MONITOR CALLS
Reference Manual

Order Number: AA-4166C-TM

January 1978

This manual describes all of the monitor calls that exist in the TOPS-20 system. For easy reference, the monitor call descriptions are arranged alphabetically and presented concisely.

This manual supersedes the manual of the same name, Order No. AA-4166B-TM.

OPERATING SYSTEM AND VERSION: TOPS-20 V03

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, February 1976
Revised: August 1976
Revised: May 1977
Revised: January 1978

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

CONTENTS

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

PREFACE

The DECSYSTEM-20 Monitor Calls Reference Manual describes all of the calls that exist in the TOPS-20 system. The manual is written for the assembly language user who is already familiar with the set of monitor calls.

Chapter 1 introduces the conventions and types of arguments used with the monitor calls. Chapter 2 presents the calls in a functional order. Chapter 3 contains the alphabetical descriptions of all the monitor calls. Appendix A is a listing of the system file, MONSYM.MAC, which is the definition of the symbols used in this manual. Appendix B is a listing of the system file, ACTSYM.MAC, which is the definition of the macros and symbols used with the USAGE monitor call.

To understand how to use the basic monitor calls, the user is referred to the DECSYSTEM-20 Monitor Calls User's Guide (DEC-20-OMUGA-A-D). This manual is organized according to the functions an assembly language user can request of the monitor. It is not a reference manual nor is it complete documentation on the entire set of monitor calls.

CHAPTER 1

INTRODUCTION

The DECSYSTEM-20 Monitor Calls Reference Manual describes every monitor call in the TOPS-20 system. These calls invoke the TOPS-20 monitor through the JSYS instruction (op code 104). The UUO-type monitor calls (op codes 40-77) invoke the compatibility package, which simulates the action of these UUO's in the TOPS-10 monitor. However, programs being written for the DECSYSTEM-20 should not be written with UUO's.

For easy reference, monitor call descriptions in Chapter 3 are arranged alphabetically and presented concisely. This concise format begins with the JSYS name and numeric definition, followed by a brief description of the JSYS function. The calling sequence for the JSYS is next, indicated by statements in the format

```
ACCEPTS IN ACn:  description
```

where n is an accumulator number. Following the list of numeric accumulators, and a description of their contents, are statements of the form

```
RETURNS     +1:  condition
            +2:  condition
```

that define where control returns, and under what conditions, after execution of the JSYS. The statement RETURNS+1: means control returns to the calling location plus 1; RETURNS+2: means control returns to the calling location plus 2. Next, there is an optional description of the action taken by the JSYS. A list of possible error mnemonics follows the JSYS definition.

The monitor calls follow the conventions described in the paragraphs below.

Arguments for the JSYS are placed in accumulators (ACs). The first argument is in AC1, the second in AC2, and so forth, up to a maximum of four accumulators. If more than four accumulators are needed by a JSYS, the arguments are in an argument block pointed to by an AC. (There are several exceptions to this convention; refer to the individual descriptions in Chapter 3.)

As mentioned above, after execution of the JSYS, control is returned to the caller at one of two locations. The +1 return is often used to indicate failure of the JSYS to perform its intended function, and an error code is stored (and available via the GETER or ERSTR call) to indicate the exact cause of the failure. This error code is usually stored in the right half of AC1. The +2 return is used to indicate successful completion of the JSYS.

However, some JSYS's have only a single return (+1) to the instruction following the call. On successful completion of the call, that instruction is executed. When an error occurs during execution of the call, the monitor examines that instruction. If it is a JUMP instruction with the AC field being either 16 or 17, the monitor transfers control to the address specified in the JUMP instruction. If the instruction following the call is not a JUMP instruction, the monitor generates an illegal instruction software interrupt, which the caller can process via the software interrupt system. If it is not prepared to process the interrupt, the process is usually terminated and a message is output. (Refer to Section 2.5.)

To include a JUMP instruction, the programmer can place one of the following symbols as the instruction following the call:

```
ERJMP address
ERCAL address
```

These symbols correspond to JUMP 16, and JUMP 17, respectively, which are machine no-ops. However, the monitor simulates a

```
JUMPA address
```

instruction when an ERJMP is used. This transfers control permanently, and the routine at the address specified must include a JRST instruction to return. The monitor simulates a

```
PUSHJ 17, address
```

instruction when an ERCAL is used. This is a subroutine call, and the routine must include a RETURN instruction to return. Note that ERCAL requires accumulator 17 to be set up as a pushdown pointer. These symbols (usually defined by OPDEFs) allow the programmer to process an error without using the software interrupt system and, in fact, override the interrupt system.

The ERJMP or ERCAL symbol can be used with all JSYS's, independent of whether the JSYS has one or two returns. The ERJMP or ERCAL will be "executed" only if it is the next instruction following a JSYS that fails; otherwise, it is a no-op.

Refer to the JSYS's descriptions in Chapter 3 for specifics on the returns possible from each JSYS.

1.1 SOURCE/DESTINATION DESIGNATORS

Many monitor calls operate on or transmit byte streams. The source or destination of these bytes can be any one of several items, including a file, a terminal, or a string in the caller's address space. In these cases, a standard 36-bit quantity, called the source/destination designator, is used as a JSYS argument to declare the byte stream on which to operate. It can have one of the following formats.

Symbol	Left Half	Right Half	Meaning
-	0	JFN	a job file number. Typically a small number starting at 1. The JFN is the job's global handle on a file and is assigned with the GTJFN monitor call. (Refer to Section 2.2.3.)
.PRIIN	0	100	primary input designator(1)
.PRIOU	0	101	primary output designator(1)
.NULIO	0	377777	null designator
.TTDES	0	400000	universal terminal designator
.CTTRM	0	777777	the job's controlling terminal
.DVDES	600000	000000	universal device designator
	reasonable left half of byte ptr	effective adr	a byte pointer to the beginning of a string
	777777	effective adr (<777777)	implicit byte pointer with left half to be changed to 440700. (Refer to Section 1.1.3.)
	777777	777777	universal default (i.e., -1)
	5xxxxx	xxxxxx	immediate numeric value

1. These designators are legal wherever a JFN is expected and cannot be assigned by the user; that is, when the user is obtaining JFNs via GTJFN (or GNJFN), 100 and 101 are never assigned.

The most commonly-used source/destination designators are:

1. A JFN identifying a particular instance of a file. This JFN will have previously been obtained from another monitor call (GTJFN JSYS, refer to Section 2.2.3).

2. The primary input and output designators. (Refer to Section 2.2.9.) These designators are the recommended ones to use to refer to the logical controlling terminal because they can be changed to cause "terminal" input and/or output to be taken from and/or sent to a file. The controlling terminal designator (0,-1) cannot be redirected in this way, and its use is not recommended in normal situations.

3. A byte pointer to the beginning of the string being read or written in the calling program's address space.

1.1.1 File Designator

A file designator is a subset of the source/destination designator. It includes all options except byte pointers to strings and numeric values.

1.1.2 Device Designator

Many monitor calls dealing with devices (refer to Section 2.4) take a device designator as an argument. This designator can be either

LH: .DVDES(600000)+device type number
RH: unit number for devices that have units, arbitrary code for structures, or -1 for non-structure devices that do not have units

or

LH: 0
RH: .TTDES(400000)+ terminal number, or .CTTRM(-1) for controlling terminal

Thus, terminals can be represented in two ways; the second way is provided for compatibility with the source/destination designator.

Because designators for structures contain an arbitrary code, these designators must always be obtained from the monitor (via the STDEV call) and cannot be created by the program.

Section 2.4 describes the various devices and their type numbers.

1.1.3 Byte Pointers To Strings

Many monitor calls deal specifically with ASCII strings. The following conventions apply to such strings.

1. A file designator can be used if the file is in 7-bit ASCII format. This is the usual format for text files.

2. One of the following ways is used to designate a string in the caller's address space:

   a. -1,, ADR to designate a 7-bit ASCII string beginning in the leftmost byte of ADR. This is for convenience, making HRROI 1,ADR functionally equivalent to MOVE 1,[POINT 7,ADR].

   b. A full word byte pointer with a byte size of 7 bits. If the byte size is not 7 bits, the results may be incorrect because the byte string is referenced via ILDB/IDPB with no additional checking.

Normally, ASCII strings are assumed to be terminated with a byte of 0 (i.e., ASCIZ). A few calls terminate on other ASCII characters because of context (e.g., NIN call), and some optionally accept an explicit byte count and/or terminating byte. These latter calls (e.g., SIN and SOUT calls) are generally those that can handle non-ASCII strings and byte sizes other than 7 bits.

After a JSYS is used to read a string, the source byte pointer argument is updated such that an ILDB would read the character following the terminating character; an LDB would reread the terminating character.

After a JSYS is used to write a string, the destination byte pointer argument is updated to point to the character following the last nonnull character written. If there is room, a null byte is appended to the string, but the byte pointer returned is such that an IDPB will overwrite the null.

1.1.4 Special Designators

The universal default designator of -1 is used to indicate the current designator, such as the current job or the connected directory. For example, the GETJI monitor call accepts an argument of -1 as the designator for the current job.

The designator 5xxxxx xxxxxx (where an immediate numeric value is in bits 3-35) is used to supply a numeric designator as an argument to a call. For example, the CACCT monitor call accepts an account number as 5B2+33-bit number.

1.2 PROCESS HANDLE

Several monitor calls accept an 18-bit argument called a process handle.

Value	Symbol	Meaning
400000	.FHSLF	current process
-1	.FHSUP	superior process
-2	.FHTOP	top-level process
-3	.FHSAI	current process and all of its inferiors
-4	.FHINF	all of the current process' inferiors
-5	.FHJOB	all processes in the job

Use of the superior process argument (.FHSUP) is legal only if the process has the superior process access capability (SC%SUP) enabled in its capability word. Use of the top level process argument (.FHTOP) is legal only if the process has the WHEEL or OPERATOR capability enabled (SC%WHL or SC%OPR) in its capability word. Refer to Section 2.6.1 for information on the capability word.

A process handle may also be in the range 400001 to 400777. These process handles are called relative process handles and are generated by the monitor (refer to the CFORK monitor call description) to refer to specific processes. These handles are valid only within the context of the process to which they are given. Thus, they may not be meaningfully passed between processes.

1.3 PROCESS/FILE HANDLE

Some monitor calls accept an 18-bit argument called a process/file handle. This handle is either a process handle (as defined in Section 1.2) or a JFN. Note that string pointers and terminal identifiers cannot be used in this context. This is not a limitation, however, because the operations that use the process/file handle are used for changing page maps. Such operations are not meaningful for string pointers or terminals.

1.4 DATE AND TIME STANDARDS

In TOPS-20, day 1 is November 18, 1858. This is consistent with the Smithsonian Astronomical Date Standard which is used by several computer systems. The date uniformly increases by one for each day of the week since day 1. Internal dates are the number of days since November 17, 1858 . Refer to Section 2.8.2 for more information about the date and time format.

The internal time of day standard for TOPS-20 is specified as a fraction of a day.

The internal date and time format is a 36-bit quantity, which frequently occurs as a JSYS argument or is returned as a value. In these cases, the day is in the left half, and the fraction of the day is in the right half.

The monitor calls that convert dates and times change from local dates and times to internal dates and times and vice versa.

1.5 CONVENTIONS USED IN THIS MANUAL

1.5.1 Number Bases

Except where otherwise noted, numbers used in this manual, including those in the definition of a JSYS description, are octal. When indicated, bits in words are numbered in decimal with the leftmost bit of the word labeled B0 and the rightmost bit of the word labeled B35.

1.5.2 Abbreviations

The following abbreviations are used in this manual:

B0, B1, ...	Bit 0, bit 1, ... of the computer word
nBm	Field whose rightmost bit is m and whose value is n (e.g., 5B2).
LH	Left Half (B0-B17 of the word)
RH	Right Half (B18-B35 of the word)
JFN	Job File Number
PSB	Process Storage Block (a table containing all monitor data for the process)
JSB	Job Storage Block (a table containing all monitor data relevant to the job)
CCOC words	Control Character Output Control words (2 words containing 36 2-bit bytes that determine the way in which control characters are output. Refer to Section 2.4.4.2.)
FDB	File Descriptor Block. Refer to Section 2.2.8.

1.5.3 Symbols

The symbols used in this manual, including the names of the JSYS's, are defined in the system file MONSYM.MAC. The user must include in his program the statement

```
SEARCH MONSYM
```

before the first occurrence of a symbol to have the symbols defined in his assembly. Refer to Appendix A for a listing of MONSYM.

1.5.4 Unimplemented Features

The MONSYM file contains symbol names for several JSYS's and bit positions that are not described in this manual. These features are not implemented in the DECSYSTEM-20. If an unimplemented JSYS is used in a user program, it causes an illegal instruction interrupt unless followed by an ERJMP or ERCAL symbol. In this case, the ERJMP will be "executed." It is recommended that unimplemented or undefined bit positions be zero to allow for future expansion.

CHAPTER 2

FUNCTIONAL ORGANIZATION OF JSYS'S

2.1 ACCOUNTING FUNCTIONS

The monitor calls in this group are used to initiate and delete jobs from the system as well as to change and read accounting information about these jobs.

The monitor calls pertaining to accounting functions are as follows:

LOGIN	Logs a job into the system
CRJOB	Creates a job and optionally logs it in
LGOUT	Kills a job
CACCT	Changes a job's account
GACCT	Reads a job's account
SACTF	Sets a file's account
GACTF	Reads a file's account
USAGE	Writes entries into the system's accounting data file
VACCT	Validates an account

2.2 REFERENCING FILES

All files in the system, including the system's file directory, are normally referenced with the calls in this group. The privileged calls for directly referencing the disk are described in Section 2.9.

2.2.1 File Specifications

A file in TOPS-20 is identified by its device name, directory name, filename, file type, and generation number. These five items uniquely identify any file on the system that is accessible to a user. The device name identifies the device on which the file is stored. The directory name identifies the directory containing the file. The filename, type, and generation number identify a particular file in the directory given by the device and directory name.

A file can also have attributes associated with it to further specify information about the file. The syntax of an attribute is any combination of the following:

1. ;T

2. ;Pprotection

3. ;Aaccount

These attributes generally apply to disk files and are not available or useful with nondisk files. The ;T attribute indicates that a file is temporary and causes bit FB%TMP to be set in the file's File Descriptor Block (FDB). The ;P attribute followed by a protection code specifies the protection of the file. This protection code is placed in word .FBPRT of the FDB. The ;A attribute followed by an account descriptor specifies the account for the file. A pointer to this account is placed in word .FBACT of the FDB.

The general format of a file specification is:

dev:<directory>name.typ.gen;attributes

Refer to the DECSYSTEM-20 User's Guide for the complete description of file specifications.

If a field of the specification is omitted, it may be supplied by the program or from standard system values. (Refer to Section 2.2.3.)

Recognition is done on file specifications in a uniform manner regardless of the source of input. However, the program can control certain aspects of recognition. Whenever an ESC is encountered in the file specification string, the previous input is looked up according to the field currently being input. A match is indicated if the input string either exactly matches an entry in the appropriate table or is an initial substring of exactly one entry. In the latter case, the portion of the matching entry not appearing in the input string is output to a specified output file. The field terminator is output also. Recognition is done on successive fields with the fields being defaulted if need be. If the file specification cannot be uniquely determined, as many entire fields as are unique are recognized, and a bell is output signifying that more input is required from the user. If the input string cannot possibly match any existing file specification, an error is returned.

CTRL/F behaves like ESC except recognition stops after the current field. This allows the filename to be recognized, for example, but not the file type.

If recognition is not used, then each field must be delimited as indicated in the general format above. The input must exactly match some existing file specification unless the program specifies in the GTJFN call that new specifications are allowed (i.e., an output file). Without ESC or CTRL/F, no recognition is done, but completely omitted fields are defaulted if the program has specified default values. The file specification is complete whenever all fields have been recognized or a terminator has been input. File specification terminators are described in the GTJFN call description.

The following editing characters are recognized during the input of file specifications:

DELETE	erases one character. If no more characters remain in the input, a bell is output.
CTRL/W	deletes back to the last punctuation character. If no more characters remain in the input, a bell is output.
CTRL/U	aborts the entire filename-gathering operation.
CTRL/R	retypes the entire input as specified so far and awaits further input.

2.2.2 Logical Names

Logical names are user-specified default values for one or more fields in a file specification. Through the use of logical names, the user can override standard file specification fields built into TOPS-20 programs because logical name fields take precedence over default fields set by a program. However, the user can still specify any fields explicitly since a logical name defines values to be used only if none are given by the user. The user defines logical names with the DEFINE command or the CRLNM monitor call. Refer to the DECSYSTEM-20 User's Guide for the complete description of logical names.

Recognition can be performed on a logical name by using ESC or CTRL/F. When recognition is in effect for the first field of the string being entered to the GTJFN call, the list of user-defined logical names is scanned first. If the data entered by the user is sufficient to uniquely define one of his logical names, then the logical name is selected. The remaining characters of the logical name and the terminating colon are output to the user. If the data entered by the user for the first field does not define a logical name, then the directory is searched for a unique filename match. Note that logical names are not searched in conjunction with directory entries and that a unique logical name will be recognized even though there may be a filename with the same beginning character string.

There is one exception to the recognition procedure described above: if an ESC or CTRL/F is typed as the first character, then the table of logical names will not be searched. This allows programs to define defaults that can be used by typing only the ESC and that cannot be circumvented if the user has only one logical name defined.

2.2.3 File Handles

It is necessary to have file handles that can be contained in a few bits and do not require extensive lookup procedures for each reference. The file specification is the fundamental handle on a file, but this specification fits neither criterion above. Therefore in TOPS-20, files are referenced by handles called JFNs (Job File Numbers). The JFN is a small number and is valid within the context of the job (i.e., within any process of the job to which it is assigned). However, the handle is not valid between jobs. That is, JFN 2 in job 11 will generally be a handle on a completely different file than JFN 2 in job 18.

A JFN is associated with a file with either the GTJFN or GNJFN monitor call. The GTJFN call accepts a file specification and returns a JFN for the indicated file. If a field of the specification is omitted, it may be supplied by the program defaults or from standard system values. If the file specification refers to a group of files (because of wildcard characters, see below), the GNJFN call can be used to associate the JFN to the next file in the group.

A logical name can apply to one or more fields of the file specification passed to the GTJFN call. The logical name must be the first identifier passed to GTJFN and must be terminated with a colon.

The GTJFN call uses a certain search order when obtaining a field in a file specification. This order is as follows:

1. Use the field explicitly typed by the user or the one specified in the primary input string.

2. Use the value for the field that is specified in the logical name specification.

3. Use the value for the field that is specified in the default block by the program. This is only for the long form of the GTJFN call.

4. Use the system default value if all of the above searches fail.

In the special case of a device field specification where the device name has been obtained from either the program default or the system default, the device field is checked to see if it is actually a logical name. If it is, then the values specified in its definition become defaults for all fields including the device field.

If the specific call to GTJFN permits, a wildcard character (either an asterisk or a percent sign) can appear in the device, directory, filename, type, or generation number fields. (The percent sign cannot appear in the generation number field.) An asterisk matches any occurrence of the field, including a null field. An asterisk as part of a field matches any remaining characters in the field, including a null character. A percent sign matches any single existing character in the field. Upon completion of the GTJFN operation, the JFN returned will reference the first file found when scanning the directory in the following order:

In numerical order by directory number
In alphabetical order by filename
In alphabetical order by file type
In ascending numerical order by generation number

The GNJFN call can then be given to associate the JFN to the next file. Normally, a program accepting wildcard characters in file specifications will successively reference all of a class of files using the same JFN and not obtain a new JFN for each file.

The maximum number of JFNs allowed is dependent upon the space reserved for JFN-related information in the Job Storage Block (JSB). Currently the maximum number of JFNs allowed is 140 (octal).

JFNs 100 (.PRIIN) and 101 (.PRIOU) are reserved for the primary input and output designators, respectively, and are never returned by the GTJFN (or GNJFN) call. JFN 377777 (.NULIO) is reserved for the null designator.

2.2.4 File References

All file operations are initiated by acquiring a JFN on a file using the GTJFN (or GNJFN) call. Some file operations, such as deleting, renaming, and status queries about the file, may be performed immediately after the JFN is acquired. Certain operations, particularly data transfers, require that the file be opened with an OPENF call on the JFN. When the user opens a file, he specifies the byte size to be used for byte I/O operations and the access requested to the file. Several implicit initialization operations, which affect subsequent references to the file, are also invoked when a file is opened. For example, a file's position pointer is normally reset to the beginning of the file such that the first sequential input operation reads the beginning data of the file.

2.2.5 Sample Program

A sample program follows which acquires JFNs, opens both an input and an output file, and then copies data from the input file to the output file in 7-bit bytes until the end of the input file is encountered.

```
;*** PROGRAM TO COPY INPUT FILE TO OUTPUT FILE. ***
;       (USING BIN/BOUT AND IGNORING NULL'S)

        TITLE FILEIO            ;TITLE OF PROGRAM
        SEARCH MONSYM           ;SEARCH SYSTEM JSYS-SYMBOL LIBRARY

;*** IMPURE DATA STORAGE AND DEFINITIONS ***


INJFN:  BLOCK 1                 ;STORAGE FOR INPUT JFN
OUTJFN: BLOCK 1                 ;STORAGE FOR OUTPUT JFN

        PDLEN=3                 ;STACK HAS LENGTH 3
PDLST:  BLOCK PDLEN             ;SET ASIDE STORAGE FOR STACK

A==1                            ;JSYS AC'S
B==2
C==3
D==4
T1==5                           ;TEMPORARY AC'S
                                ;....
P==17                           ;PUSH DOWN POINTER

;*** PROGRAM INITIALIZATION ***

START:  RESET                   ;CLOSE FILES AND INITIALIZE PROCESS
        MOVE P,[IOWD PDLEN,PDLST] ;ESTABLISH STACK

;*** GET INPUT-FILE ***

INFIL:  HRROI A,[ASCIZ /
INPUT FILE: /]                  ;PROMPT FOR INPUT FILE
        PSOUT                   ;ON CONTROLLING  TERMINAL
        MOVE A,[GJ%OLD+GJ%FNS+GJ%SHT];SEARCH MODES FOR GTJFN
                                ;[EXISTING FILE ONLY , FILE-NR'S IN B
                                ; SHORT CALL ]

        MOVE B,[.PRIIN,,.PRIOU] ;GTJFN'S I/O WITH CONTROLLING TERMINAL
        GTJFN                   ;GET JOB FILE NUMBER (JFN)
         ERCAL [ PUSHJ P,WARN   ;IF ERROR, GIVE WARNING
                JRST INFIL]     ;AND LET HIM TRY AGAIN
        MOVEM A,INJFN           ;SUCCESS, SAVE THE JFN
```

```
;*** GET OUTPUT-FILE ***

OUTFIL: HRROI A,[ASCIZ /
OUTPUT FILE: /]                 ;PROMPT FOR OUTPUT FILE
        PSOUT                   ;PRINT IT
        MOVE A,[GJ%FOU+GJ%MSG+GJ%CFM+GJ%FNS+GJ%SHT];GTJFN SEARCH MODES
                                ;[DEFAULT TO NEW GENERATION , PRINT
                                ; MESSAGE , REQUIRE CONFIRMATION
                                ; FILE-NR'S IN B , SHORT CALL ]

        MOVE B,[.PRIIN,,.PRIOU] ;I/O WITH CONTROLLING TERMINAL
        GTJFN                   ;GET JOB-FILE NUMBER
         ERCAL [ PUSHJ P,WARN   ;IF ERROR, GIVE WARNING
                JRST OUTFIL]    ;AND LET HIM TRY AGAIN
        MOVEM A,OUTJFN          ;SAVE THE JFN


;NOW, OPEN THE FILES WE JUST GOT

;    INPUT

        MOVE A,INJFN            ;RETRIEVE THE INPUT JFN
        MOVE B,[7B5+OF%RD]      ;DECLARE MODES FOR OPENF [7-BIT BYTES + INPUT]
        OPENF                   ;OPEN THE FILE
         ERJMP FATAL            ;IF ERROR, GIVE MESSAGE AND STOP

;    OUTPUT

        MOVE A,OUTJFN           ;GET THE OUTPUT JFN
        MOVE B,[7B5+OF%WR]      ;DECLARE MODES FOR OPENF [7-BIT BYTES + OUTPUT]
        OPENF                   ;OPEN THE FILE
         ERJMP FATAL            ;IF ERROR, GIVE MESSAGE AND STOP


;*** MAIN LOOP :COPY BYTES FROM INPUT TO OUTPUT ***

LOOP:   MOVE A,INJFN            ;GET THE INPUT JFN
        BIN                     ;TAKE A BYTE FROM THE SOURCE
         ERJMP FATAL            ;IF ERROR, GIVE MESSAGE AND STOP
        JUMPE B,DONE            ;IF 0, CHECK FOR END OF FILE.
        MOVE A,OUTJFN           ;GET THE OUTPUT JFN
        BOUT                    ;OUTPUT THE BYTE TO DESTINATION
         ERJMP FATAL            ;IF ERROR, GIVE MESSAGE AND STOP
        JRST LOOP               ;LOOP, STOP ONLY ON A 0 BYTE (FOUND
                                ;AT LOOP+2)

;*** TEST FOR END OF FILE, ON SUCCESS FINISH UP ***

DONE:   GTSTS                   ;GET THE STATUS OF INPUT FILE.
        TLNN B,(GS%EOF)         ;AT END OF FILE?
        JRST LOOP               ;NO, FLUSH NULL AND CONTINUE COPY

CLOSIF: MOVE A,INJFN            ;YES, RETRIEVE INPUT JFN
        CLOSF                   ;CLOSE INPUT FILE
         ERJMP FATAL            ;IF ERROR, GIVE MESSAGE AND STOP

CLOSOF: MOVE A,OUTJFN           ;RETRIEVE OUTPUT JFN
        CLOSF                   ;CLOSE OUTPUT FILE
         ERJMP FATAL            ;IF ERROR, GIVE MESSAGE AND STOP
        HRROI A,[ASCIZ/
[DONE]/]                        ;SUCCESSFULLY DONE
        PSOUT                   ;PRINT IT
        JRST ZAP                ;STOP
```

```
;*** ERROR HANDLING ***

FATAL:  HRROI A,[ASCIZ/
?/]                             ;FATAL ERRORS PRINT ? FIRST
        PUSHJ P,ERROR           ;THEN PRINT ERROR MESSAGE,
        JRST ZAP                ;AND STOP

WARN:   HRROI A,[ASCIZ/
%/]                             ;WARNINGS PRINT % FIRST
                                ; AND FALL THRU 'ERROR' BACK TO CALLER

ERROR:  PSOUT                   ;PRINT THE ? OR %
        MOVE A,[.PRIOU]         ;DECLARE PRINCIPAL OUTPUT DEVICE FOR ERROR MESSAGE
        MOVE B,[.FHSLF,,-1]     ;CURRENT FORK,, LAST ERROR
        SETZB C,D               ;NO LIMIT,, FULL MESSAGE
        ERSTR                   ;PRINT THE MESSAGE
         JFCL                   ;IGNORE UNDEFINED ERROR NUMBER
         JFCL                   ;IGNORE ERROR DURING EXECUTION OF ERSTR
        POPJ P,                 ;RETURN TO CALLER

ZAP:    HALTF                   ;STOP
        JRST START              ;WE ARE RESTARTABLE
        END START               ;TELL LINKING LOADER START ADDRESS
```

2.2.6 File Access

TOPS-20 provides a general mechanism for protecting files against unauthorized access. Generally, access to a file depends on the kind of access desired and the relationship of the user making the access to the directory containing the file. The possible relationships a user may have to the file's directory are:

1. The directory containing the file is the user's connected or accessed directory. Users satisfying this relationship have owner access to the files in the directory.

2. The directory containing the file is in the same group as the user. Users satisfying this relationship have group member access to the files in the directory.

3. The directory containing the file is outside the group membership. Users satisfying this relationship have world access to the files in the directory.

Both users and directories may belong to groups. The group-member relationship is satisfied if both the directory and the user belong to one or more of the same groups. Groups are assigned by the system manager or operator. (Refer to the DECSYSTEM-20 Operator's Guide.)

The type of access permitted to a file for each relationship is represented by the value of a 6-bit field. The possible values are:

Value	Symbol	Meaning
40	FP%RD	Read access
20	FP%WR	Write access
4	FP%APP	Append access
2	FP%DIR	Directory listing access. A GTJFN call for the file will fail if the user does not have at least this access.

This 6-bit field and the three relationships (owner, group, remaining users) are represented by an 18-bit code, with bits 0-5 being the owner, bits 6-11 being the group, and bits 12-17 being the remaining users. When a particular bit is on, the corresponding access is permitted for the particular relationship.

The access given to a group member includes the access given to all members outside the group. Also, the access given to the owner includes the access given to group members. Thus, the owner of a file or a user in the owner's group cannot have less access than users outside the group.

2.2.7 Directory Access

Access to a directory is protected in a manner similar to, but distinct from, that of a file. There is an 18-bit code, containing three 6-bit fields, associated with each directory. Each of the three fields controls access by users in the same way that access to files is controlled. For directories, however, each 6-bit field can have one of the following values.

Value	Symbol	Meaning
40	DP%RD	Accessing files in the directory according to the access code on the individual files is allowed. A GTJFN call for a file in the directory will fail if the user does not have this access.
10	DP%CN	Connecting to the directory without giving a password is allowed. With this access, a group member can change the FDB (as the owner) as well as times, dates, and accounting information for files in the directory. Other operations on the files are subject to the access codes of the files. If the user is connected to the directory, he has ownership access to the files; if he is not connected, he has group membership access.
4	DP%CF	Creating files in the directory is allowed.

When a user requests access to a file, the monitor checks the directory access code first. If the directory code allows the desired access, the monitor then checks the access code of the individual file.

The access actually granted to a file is specified when the user opens the file with the OPENF call. If the access specified in the OPENF call is the same as the access permitted by the 18-bit access code, the user is granted access to the file. Thus, for a user to be granted access to a specific file, two conditions must be met:

1. The access code (both directory and file) must permit the user to access the file in the desired manner (e.g., read, write).
2. The file must not be open for a conflicting type of access.

2.2.8 File Descriptor Block

Each file has an associated File Descriptor Block (FDB). The FDB can be read with the GTFDB call, and selected words of this block can be changed with the CHFDB call. The name strings of the author and last user who wrote to the file can be read with the GFUST call and changed with the SFUST call.

The format of the FDB is shown in Table 2-1. Words that can be changed with the CHFDB call are indicated by the listing of the minimum capability (refer to Section 2.6.1) required to change the word.

Table 2-1
File Descriptor Block (FDB)

Word	Symbol	Meaning	
0	.FBHDR	Header word	
1	.FBCTL	B0(FB%TMP)	File is temporary. This bit can be changed by owner.
		B1(FB%PRM)	File is permanent. This bit can be changed by owner.
		B2(FB%NEX)	File does not have a file type yet; file does not really exist. This bit cannot be changed.
		B3(FB%DEL)	File is deleted. This bit can be changed by owner.
		B4(FB%NXF)	File does not exist because it has not yet been closed. This bit cannot be changed.
		B5(FB%LNG)	File is longer than 512 pages. This bit cannot be changed.
		B6	Reserved for DEC.
		B7(FB%DIR)	File is a directory. This bit cannot be changed.
		B8(FB%NOD)	File is not to be saved by the backup system. This bit can be changed by owner or user with write access.
		B9(FB%BAT)	File may have one or more bad pages. This bit cannot be changed.
		B10(FB%SDR)	Directory has subdirectories. This bit cannot be changed.

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
1	.FBCTL (Cont.)	B14-B17 (FB%FCF) File class field. This field can be changed by owner. If value of field is 0(.FBNRM), file is not an RMS file. If value of field is 1(.FBRMS), file is an RMS file.
2	.FBEXL	Link to FDB of next file with the same name but different file type. This word cannot be changed.
3	.FBADR	Disk address of file index block. This word cannot be changed.
4	.FBPRT	File access code. LH: 500000 RH: file access bits. This field can be changed by owner.
5	.FBCRE	Date and time of last write to the file. Is modified when any program writes to the file. This word can be changed by a process with OPERATOR capability enabled.
6	.FBAUT	Pointer to string containing the name of the author. This name is read with the GFUST monitor call and can be changed with the SFUST monitor call.
7	.FBGEN	Generation and directory numbers of file. LH(FB%GEN): generation number of the file. RH(FB%DRN): monitor internal directory number of the file (only if B7 of .FBCTL is on). This word cannot be changed.
10	.FBACT	Account information. This word is either a pointer to an alphanumeric account designator or 5B2 + numeric account designator. This word can be changed with the SACTF monitor call.
11	.FBBYV	File I/O information. B0-B5(FB%RET) Number of generations to retain (retention count). If two generations of the same file have different retention counts, the count is taken from the generation currently being used. This field can be changed by owner.

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
11	.FBBYV (Cont.)	B6-B11(FB%BSZ) File byte size. This field can be changed by user with write access. B14-B17(FB%MOD) Data mode of last open of file. This field can be changed by user with write access. B18-B35(FB%PGC) Page count of file. This field can be changed by a process with WHEEL or OPERATOR capability enabled.
12	.FBSIZ	Number of bytes in the file. (Refer to Section 2.2.11.) This word can be changed by user with write access.
13	.FBCRV	Date and time of creation of file. This word can be changed by user with write access.
14	.FBWRT	Date and time of last user write to file. Is modified only by the user. This word can be changed by user with write access.
15	.FBREF	Date and time of last nonwrite access to file. This word can be changed by user with write access.
16	.FBCNT	Count word. LH: number of writes to file. RH: number of references to file. This word can be changed by a process with OPERATOR capability enabled.
17-23	.FBBK0- .FBBK4	Five words for backup system. The first word is used by the DUMPER program; remaining words are reserved for the future.
24	.FBUSW	User settable word. This word can be changed by owner.
25	.FBGNL	Address of FDB for next generation of file. This word cannot be changed.

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
26	.FBNAM	Pointer to filename block. This word cannot be changed.
27	.FBEXT	Pointer to file type block. This word cannot be changed.
30	.FBLWR	Pointer to string containing the name of the user who last wrote to the file. This name is read with the GFUST monitor call and can be changed with the SFUST monitor call.
31	.FBLEN	Length of the FDB. This word cannot be changed.

2.2.9 Primary Input And Output Files

Each process in a job has a primary input file and a primary output file. Both files are normally the controlling terminal, but can be changed to other files (with the SPJFN call). The primary input and output files are referenced with designators .PRIIN (JFN 100) and .PRIOU (JFN 101), respectively. Programs should be coded to do their "terminal" I/O to these designators, so that they can be used with command files without modification; only in extreme cases should a program reference its controlling terminal (0,,-1) directly.

2.2.10 Methods Of Data Transfer

Most I/O is done sequentially by bytes, as shown in the sample program. (Refer to Section 2.2.5.) This form of data transfer may be used with any file. A pointer maintained in the monitor is implicitly initialized when a file is opened and advanced as data is transferred. For files on disk, there are two other methods of data transfers. First, random access byte I/O is possible by using the SFPTR call or the RIN/ROUT calls. Second, entire pages of data may be mapped with the PMAP call.

2.2.11 File Byte Count

For disk files, TOPS-20 maintains a file byte count (.FBSIZ) in the FDB. This count is set by the monitor when sequential output (e.g., BOUT, SOUT) occurs to the file and thus, on sequential output, reflects the number of bytes written in the file.

When output occurs to the file using the PMAP call, the monitor does not set the file byte count. In this case, the number of bytes in the file may be different from the file byte count stored in the FDB. To allow sequential I/O to occur later to the file, the program should update the file byte count (.FBSIZ) and the file byte size (FB%BSZ) in the FDB before closing the file. The values are updated with the CHFDB monitor call.

When output occurs to the file using random output calls (e.g., ROUT), the file byte count is a number one greater than the highest byte number in the file.

The file byte count is interpreted according to the byte size stored in the FDB, not the byte size specified when the file is opened. When a new file is opened, the byte size stored in the FDB is 36 bits, regardless of the byte size specified in the OPENF call. If the program executes a CHFDB call to change the file byte count, usually it must also change the byte size (FB%BSZ) so that both values reflect the same size bytes.

2.2.12 EOF Limit

There is an EOF limit associated with every opening of a file. This limit is the number of bytes that can be read with a sequential input call (e.g., BIN, SIN). When the program attempts to read beyond this limit using sequential input, the call returns a 0 byte and an end-of-file condition. This condition may generate a software interrupt (refer to Section 2.5) if the user has not included an ERJMP or ERCAL as the next instruction following the call. (Refer to Chapter 1.)

The EOF limit is computed when the file is opened with the OPENF call. The monitor computes this limit by determining the total number of words in the file and dividing this number by the byte size given in the OPENF call. The total number of words in the file is determined from the file byte count (.FBSIZ) and the file byte size (FB%BSZ) stored in the FDB.

2.2.13 Input/Output Errors

While performing I/O or I/O-related operations, it is possible to encounter one or more error conditions. Some of these are user-caused errors (e.g., illegal access attempts), and others are I/O device or medium errors. TOPS-20 indicates such error conditions by setting error bits in the JFN status word (refer to the GTSTS call) and by initiating a software interrupt request (refer to Section 2.5) if the user has not included an ERJMP or ERCAL after the call. If the process in which an I/O error occurs is not prepared to process the interrupt, the interrupt is changed into a process terminating condition with the expectation that the process' immediate superior will handle the error condition. The TOPS-20 Command Language is prepared to detect and diagnose I/O errors; thus, a process running directly beneath the process containing the Command Language need not do its own I/O error handling unless it chooses to do something special.

I/O errors can occur while a process is executing ordinary machine instructions as well as JSYS's. For example, if a PMAP operation is performed that maps a page of a file into a page of a process, the file I/O transfer usually does not actually occur until a reference is made by the process to that particular page of the file. If there is an I/O error in the transfer, it is detected at the time of this reference.

An attempt to do I/O to a terminal that is assigned to another job (as a controlling terminal or with the ASND call) normally results in an error, but is legal if the process has the WHEEL capability enabled.

The monitor calls used in referencing files are:

GTJFN	Assigns a JFN to a file
GNJFN	Assigns a JFN to the next file
JFNS	Translates a JFN to a string
SPJFN	Sets primary JFNs
GPJFN	Returns primary JFNs
SWJFN	Transposes two JFNs
RLJFN	Releases a JFN
OPENF	Opens a file
CLOSF	Closes a file
CLZFF	Closes a process' files
PSOUT	Outputs string to primary output designator
PBIN	Reads byte from primary input designator
PBOUT	Output byte to primary output designator
BIN	Reads the next byte
BOUT	Outputs the next byte
SIN	Reads a string
SOUT	Outputs a string
SINR	Reads a record
SOUTR	Outputs a record
RIN	Reads a byte nonsequentially
ROUT	Outputs a byte nonsequentially
DUMPI	Reads data in unbuffered data mode
DUMPO	Outputs data in unbuffered data mode
PMAP	Maps pages
RSCAN	Reads and outputs rescan buffer
RDTTY	Reads data from primary input designator
TEXTI	Reads data from terminal or file
CRLNM	Creates a logical name
INLNM	Outputs logical names
LNMST	Translates logical name to string
CHFDB	Changes a File Descriptor Block
GTFDB	Reads a File Descriptor Block
SFUST	Changes the author or last writer name string
GFUST	Reads the author or last writer name string
CHKAC	Checks access to a file
ACCES	Specifies access to a directory
DIRST	Translates directory or user number to a string
RCDIR	Translates directory name to number
RCUSR	Translates user name to number
SIZEF	Obtains file's length
SFBSZ	Sets file's byte size
RFBSZ	Reads file's byte size
SFPTR	Sets file's pointer
RFPTR	Reads file's pointer
BKJFN	Backspaces file's pointer
RNAMF	Renames a file
SFTAD	Sets file's time and dates
RFTAD	Reads file's time and dates
STSTS	Sets file's status
GTSTS	Reads file's status
UFPGS	Updates file's pages

DELF	Deletes a file
DELDF	Expunges deleted files
DELNF	Retains specified number of generations of file
FFFFP	Finds first free file page
FFUFD	Finds first used file page

2.3 OBTAINING INFORMATION

The monitor calls in this group are used to obtain information from the system, such as the time of day, resources used by the current job, error conditions, and the contents of system tables.

Several of these calls return time values (e.g., intervals, accumulated times). Unless otherwise specified, these values are integer numbers in units of milliseconds.

2.3.1 Error Mnemonics And Message Strings

Each failure for a JSYS is associated with an error number identifying the particular failure. These error numbers are indicated in the manual by mnemonics (e.g., DEVX1) and are listed with the appropriate calls. Some calls return the error number in the right half of an accumulator, usually in AC1; however, all calls leave the number in the Process Storage Block for the process in which the error occurred. Thus, a process can obtain (via the GETER call) the number for the last error that occurred.

In addition to the mnemonic of six characters or less, each error number has a text message associated with it that describes the error in more detail. These messages are listed along with the mnemonics. The ERSTR call can be used to return the message string associated with any given error number. The use of this call is recommended for handling error returns.

Refer to Chapter 3 and Appendix A for the listing of the error numbers, mnemonics, and messages.

2.3.2 System Tables

The contents of several system tables are available to programs for such purposes as generating status reports and collecting system performance statistics. Each table is identified by a fixed name of up to six characters and consists of a variable number of entries. The -1 entry in each table is the negative of the number of data entries in the table; the data entries are identified by an index which increments from 0.

Two calls exist for accessing tables. The first, SYSGT, accepts a table name and returns the table length, its first data entry, and a number identifying the table. The second, GETAB, accepts the table number returned by SYSGT or obtained from the MONSYM file and returns additional entries from the table.

The system tables are as follows. Numeric table indexes are given in octal.

Name	Index		Contents
JOBTTY	Job #	LH:	controlling terminal line number, or -1 if none (i.e., job is detached)
		RH:	reserved for DEC
JOBRT	Job #	CPU	time used by the job (negative if no such job)
TTYJOB	line number	LH:	positive job number for which this is the controlling terminal, or -1 for unassigned line, or -2 for line currently being assigned, or job number to which this line is assigned.
		RH:	-1 if no process is waiting for input from this terminal; other than -1 if some process is waiting for input.

TICKPS One-word table containing number of clock ticks per second.

NCPGS One-word table containing number of pages of real (physical) user core available in system.

The device tables DEVNAM, DEVCHR, and DEVUNT are parallel in that the same entry in each table pertains to the same device.

DEVNAM SIXBIT device name including unit number, e.g., MTA3

DEVCHR Device characteristics word, as described under the DVCHR JSYS in Chapter 3, except that B5 (DV%AV) is not meaningful.

DEVUNT LH: job number to which device is assigned (with ASND), or -1 if device is not assigned
RH: unit number, or -1 if device has no units (e.g., DSK:)

DSKERR Information on disk errors

0	number of recoverable disk errors
1 to n	varies depending on type of disk being used

DRMERR Information on drum errors

0	number of recoverable errors
1 to n	varies depending on type of drum being used

SYSVER An ASCII string identifying the system name, version, and date.

SYSTAT Monitor statistics. The entries in this table are as follows:

0	time with no runnable jobs
1	waiting time with 1 or more runnable jobs (waiting for page swapping)
2	time spent in scheduler
3	time spent processing pager traps
4	number of drum reads
5	number of drum writes
6	number of disk reads
7	number of disk writes
10	number of terminal wakeups
11	number of terminal interrupts
12	time integral of number of processes in the balance set

13	time integral of number of runnable processes
14	exponential 1-minute average of number of runnable processes
15	exponential 5-minute average of number of runnable processes
16	exponential 15-minute average of number of runnable processes
17	time integral of number of processes waiting for the disk
20	time integral of number of processes waiting for the drum
21	number of terminal input characters
22	number of terminal output characters
23	number of system core management cycles
24	time spent doing postpurging
25	number of forced balance set process removals
26	time integral of number of processes in swap wait
27	scheduler overhead time (same as entry 2) in high precision units
30	idle time (same as entry 0) in high precision units
31	lost time (same as entry 1) in high precision units
32	user time in high precision units

NOTE

This table is subject to change (usually additions) as measuring routines are added to the system.

QTIMES	0 to n	Accumulated runtime of jobs on the n scheduler queues
JOBNAM	Job #	LH: reserved for DEC RH: index into the system program tables for the system program being used by this job (determined by the last SETSN call executed by the job)
JOBPNM	Job #	SIXBIT name of program running in this job

The system program tables SNAMES, STIMES, SPFLTS, SSIZE, and SNBLKS are parallel in that the same entry in each table pertains to the same system program. The system program being run by a specific job may be determined from SNAMES, using an index obtained from table JOBNAM (above).

SNAMES SIXBIT name of system program, or 0 if this entry is unused in this and the corresponding four tables.

STIMES Total runtime of system program

SPFLTS Total number of page faults of system program

SSIZE Time integral of working set size

SNBLKS Number of samples in working set size integral

DBUGSW Debugging information
- 0 state of operator coverage (0=unattended, 1=attended, 2=debugging)
- 1 state of BUGCHK handling (0=proceed, 1=breakpoint)

LOGDES	Logging information	
	0	designator for logging information
	1	designator for job 0 and error information
PTYPAR	Pseudo-TTY parameter information	
	0	LH: number of PTYs in system RH: TTY number of first PTY
SYMTAB	SIXBIT table names of all GETAB tables	
DWNTIM	Downtime information	
	0	date and time when system will be shut down next
	1	date and time when system will subsequently be up
BLDTD	Date and time system was generated	
APRID	Processor serial number	
HQLAV	High queue load averages	
LQLAV	Low queue load averages	

The following monitor calls are used for obtaining information:

GETER	Returns the last error condition
SETER	Sets the last error condition
ERSTR	Translates an error number to a string
ESOUT	Returns an error string
SYSGT	Returns values for a system table
GETAB	Returns a word from a system table
SETNM	Sets the program's private name
SETSN	Sets the program's system and private names
GETNM	Returns the program name being used by the job
SETJB	Sets a job's parameters
GETJI	Returns job information for specified job
GJINF	Returns job information for current job
STAD	Sets the system's date
GTAD	Returns the system's date
TIME	Returns the time since the system was restarted
TIMER	Sets the runtime limit of a job
RUNTM	Returns the runtime of a job or process
HPTIM	Returns the high-precision clock values
GTDAL	Returns the disk allocation of a directory
GTRPI	Returns the paging trap information
GTRPW	Returns the trap words

2.4 COMMUNICATING WITH DEVICES

The monitor calls in this group are used to communicate with the devices on the system. Some of these devices are line printers, magnetic tapes, terminals, and card readers.

Many of the monitor calls in this group take a device designator as an argument. This designator can be either

LH: .DVDES(600000)+device type number
RH: unit number for devices that have units, arbitrary code for structures, or -1 for non-structure devices that do not have units

or

LH: 0
RH: .TTDES(400000)+terminal number, or .CTTRM(-1) for controlling terminal

The STDEV monitor call is used to convert a string to its corresponding device designator.

The various devices are:

Name	Description	Type	Units	Symbol
DSK:	disk structure	0		.DVDSK
MTA:	magnetic tape	2	yes	.DVMTA
LPT:	line printer	7	yes	.DVLPT
CDR:	card reader	10	yes	.DVCDR
FE:	front-end pseudo-device	11		.DVFE
TTY:	terminal	12	yes	.DVTTY
PTY:	pseudo-terminal	13	yes	.DVPTY
NUL:	null device	15		.DVNUL
NET:	ARPA network	16		.DVNET

The null device is an infinite sink for unwanted output and returns an EOF on input.

Device-dependent status bits are defined for some devices. These bits can be set or returned with the SDSTS or GDSTS call, respectively.

When an assignable device is assigned (by the ASND call) or opened (by the OPENF call) by one job, other jobs cannot

1. Assign the device with ASND.

2. Execute an OPENF call for the device, even if the JFN properly represents the device.

Structures are not restricted to these limitations; more than one user can simultaneously execute the OPENF call for files on structures.

2.4.1 Line Printer (LPT:)

The line printer normally accepts the 128 7-bit ASCII character codes (0-177 octal). However, by specifying a byte size of 8 when opening the printer, a program can transfer 8-bit bytes. Thus, the program can take advantage of printers that have more than 128 characters.

Each code sent usually causes a graphic to be printed. (Note that on a 64-character printer, lower case letters are represented as upper case.) However, the carriage control characters do not cause a graphic to be printed; instead they cause specific actions to be taken. The actions taken are determined by the translation RAM and the Vertical

Formatting Unit. These actions can be redefined by the installation, and the method by which they are redefined depends on the type of printer being used.

For the LP10 printer, which has a carriage control tape, the installation must change the tape to redefine the resulting actions.

For the LP05 and LP14 printers, which have a direct access Vertical Formatting Unit and a programmable translation RAM, the installation can redefine the resulting actions by:

1. Reprogramming the VFU by changing the VFU file with the MAKVFU program and reloading this file and the RAM.

2. Reprogramming the translation RAM by changing the RAM file with the MAKRAM program and reloading this file.

Refer to the LPINI and MTOPR monitor calls for the functions used in loading the VFU and RAM files.

The default actions taken on the carriage control characters, along with the default channels that determine these actions, are as follows:

ASCII Character Code	Default Channel	Name	Default Action
11		Tab	No vertical motion. Skips to the beginning of every 8th column on the same line.
12	8	Line feed	Skips to column 1 on the next line. The last six lines of each page are skipped.
13	7	Vertical tab	Skips to column 1 on the line at the next third of a page.
14	1	Form feed	Skips to column 1 on the top of the next page.
15		Carriage return	No vertical motion. Returns to column 1 of the current line and does not advance the paper.
20	2	Half page	Skips to column 1 on the next half page.
21	3	Alternate lines	Skips to column 1 on the next even line.
22	4	Three lines	Skips to column 1 on the next of every third line.
23	5	Next line	Skips to column 1 on the next line without skipping the last six lines on a page.
24	6	Sixth page	Skips to column 1 on the next sixth of a page.

The association between the ASCII code and the channel is determined by the RAM. The association between the channel and the default action is determined by the VFU. Therefore, a change in the VFU changes the association between the channel and the action, which causes the ASCII code to be associated with the new action.

2.4.1.1 **Status Bits** - The following device-dependent status bits are defined for the line printer. These bits can be obtained with the .MORST function of the MTOPR call.

Bit	Symbol	Meaning
0	MO%LCP	Lower case printer
10	MO%FER	Fatal hardware error. This error generates an interrupt on software channel .ICDAE (refer to Section 2.5.1).
12	MO%EOF	All data sent to the printer has actually been printed.
13	MO%IOP	I/O in progress
14	MO%SER	Software error (e.g., interrupt character, page counter overflow)
15	MO%HE	Hardware error. Forms must be realigned. This error generates an interrupt on software channel .ICDAE.
16	MO%OL	Device is off line
17	MO%FNX	Device is nonexistent
30	MO%RPE	RAM parity error
31	MO%LVU	Optical VFU
33	MO%LVF	VFU error
34	MO%LCI	Character interrupt. This generates an interrupt on channel .ICDAE.
35	MO%LPC	Page counter register overflow

2.4.2 **Magnetic Tape (MTA:)**

The following device-dependent bits are defined for the magnetic tape.

Bit	Symbol	Meaning
18	MT%ILW	Illegal write (i.e., the write-lock bit is set and the user attempted a write operation)
19	MT%DVE	Device error (hung or data late)
20	MT%DAE	Data error
21	MT%SER	Suppress automatic error recovery procedures

Bit	Symbol	Meaning
22	MT%EOF	Device EOF (file) mark
23	MT%IRL	Incorrect record length (not the same number of words as specified by the read operation or not a whole number of words)
24	MT%BOT	Beginning of tape
25	MT%EOT	End of tape
26	MT%EVP	Even parity
27-28	MT%DEN	Density (0 is system default) 1 .MTLOD low (200 BPI) 2 .MTMED medium (556 BPI) 3 .MTHID high density (800 BPI) Refer to the .MOSDN function of the MTOPR call for setting the density.
29-31	MT%CCT	Character counter if MT%IRL is on.

Data transfers to and from the magnetic tape can be performed using either buffered or unbuffered I/O.

2.4.2.1 **Buffered** I/O - The monitor uses buffered I/O when the sequential I/O calls (e.g., BIN/BOUT, SIN/SOUT) are used to read from or write to the magnetic tape. When the tape is opened for sequential I/O (data mode 0 on the OPENF call), the monitor reserves buffer space large enough to hold two records of data. The maximum size of the records is specified with the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR monitor call. The default size of the records is 1000(octal) words, and the maximum size of a record is 20000(octal) words. The buffers reserved by the monitor allow the user's program to overlap computation with the transfer of data to and from the tape.

The BIN monitor call is used to read one byte from the tape with the monitor filling one buffer with data as the user program is reading bytes from the other buffer. A program reading data from the tape with successive BIN calls obtains a stream of bytes until a tape mark is read. The SIN monitor call is used to read a specified number of bytes with the monitor again performing the double buffering. Both the BIN and the SIN calls read across record boundaries on the tape. The SINR monitor call is used to read variable-length records from the tape because each call returns one record to the user program. If the record on the tape contains more data than the SINR call request, the remaining bytes in the record are discarded. The SINR call never reads across record boundaries on the tape. Thus, each SINR call begins reading at the first byte of the next record on the tape. With all three calls, the specified record size must be at least as large as the largest record being read from the tape.

The BOUT monitor call is used to write one byte on the tape. A program writing data on the tape with successive BOUT calls writes a stream of bytes packed into records of the specified size. The SOUT monitor call is used to write a specified number of bytes into one record equal to the given record size. The SOUTR call is used to write variable-length records on the tape because each call writes at least one record. The size of the record is equal to either the number of bytes specified in the SOUTR call or the number of bytes specified in the maximum record size, whichever is smaller. If the number of bytes requested in the call is greater than the specified record size, then records of the maximum size are written, plus another record containing the remaining bytes. If the end of tape marker is reached during sequential mode output, the data is written and an error return is given. Bit MT%EOT (bit 25) in the device status word will be set to indicate this condition.

When a CLOSF monitor call is executed for a magnetic tape to which buffered output is being done, any data remaining in the monitor's buffers will be written to the tape. The monitor writes two tape marks after the last record written and backspaces over the second mark. This allows a subsequent write operation to overwrite the last tape mark, and always leaves two tape marks (a logical end of tape) after the last record written.

The monitor does not write records of less than four words long. Thus if the user requests less than four words to be written on a SOUTR or DUMPO (see below) call, the monitor will write a four-word record, completing it with zeros. On a SOUT call, if less than four words remain in the buffer at the time of the CLOSF call, the monitor again fills the record with zeros.

2.4.2.2 **Unbuffered I/O** - The DUMPI and DUMPO monitor calls are used to read from or write to the magnetic tape without using buffered I/O. (Unbuffered I/O is sometimes called dump mode.) Unbuffered I/O uses a program-supplied command list to determine where to transfer data into or out of the program's address space. The command list can contain three types of entries:

1. IOWD n, loc transfers n words from loc through loc+n-1. The next command is obtained from the location following the IOWD. Each IOWD word reads or writes a separate magnetic tape record.

2. XWD 0, y takes the next command from location y.

3. 0 terminates the command list.

Refer to the DUMPI call description for more information.

On input, a new record is read for each IOWD entry in the command list. If the IOWD request does not equal the actual size of the record on the tape, an error (IOX5) is returned. The GDSTS monitor call can then be executed to examine the status bits set and to determine the number of words transferred. In addition, if a tape mark is read, an error (IOX4) is returned. On output, a new record is written for each IOWD entry in the command list.

There are two modes available in unbuffered I/O. In the normal mode, the monitor waits for the data transfer to complete before returning control to the program. In the no-wait mode, the monitor returns control immediately after queueing the first transfer so that the program can set up the second transfer. The monitor then waits for the first transfer to complete before queueing the second. If the first transfer is successful, the second one is started, and control is returned to the program. If the first transfer is not successful, an error is returned in AC1, and the second one is not started. The desired mode is specified by bit DM%NWT in AC1 on the DUMPI or DUMPO call.

2.4.2.3 **Magnetic Tape Status** - The status word of a magnetic tape can be obtained with the GDSTS call or individual status bits can be obtained with the MTOPR call. The GDSTS call waits for all activity to stop during sequential mode output, dump mode, and spacing operations before obtaining the status. A GDSTS call executed during sequential mode input returns the status of the current record.

Reading from or writing to a magnetic tape cannot be done if there are any errors set in the device status word. The program can clear errors with the SDSTS call or the .MOCLE function of the MTOPR call.

2.4.2.4 **Reading A Tape in the Reverse Direction** - With the .MOSDR function of the MTOPR call, the program can cause the tape to move in the reverse direction (toward the beginning of the tape) during read operations. The data in each record is returned in the forward order, but the records themselves are returned in the reverse order. The beginning of tape mark is considered a tape mark.

When the SIN call is used to read data in the reverse direction, it is recommended that the number of bytes requested by the call and the record size be equal to the size of the record on the tape. This returns the bytes in the entire record in the correct order, but returns the records in reverse order. If the request does not equal the sizes, the bytes returned will be out of phase with the bytes in the actual record.

When the SINR call is used to read data in the reverse direction, the number of bytes requested by the call should be at least as large as the size of the record on the tape. If the requested number is smaller than the actual record, the remaining bytes in the record are discarded from the beginning of the record and not from the end of the record.

2.4.2.5 **Hardware Data Modes** - By using the .MOSDM function of the MTOPR call, the program can set the mode for storing data on a magnetic tape. The following descriptions indicate how bits are stored in the tracks and the number of frames required to store a 36-bit word of data. Note that the diagrams are logical representations of the stored data. In reality, the parity frame (P) is in the center of the tape, and the order of the remaining frames may not be as shown.

Unbuffered (Dump) Mode

This mode stores a word of data as a 36-bit byte in five frames of a 9-track tape. Note that the fifth frame is partially used. This mode is normally the default mode.

TRACKS

9	8	7	6	5	4	3	2	1	FRAMES
B0	B1	B2	B3	B4	B5	B6	B7	P	
B8	B9	B10	B11	B12	B13	B14	B15	P	
B16	B17	B18	B19	B20	B21	B22	B23	P	
B24	B25	B26	B27	B28	B29	B30	B31	P	
0	0	0	0	B32	B33	B34	B35	P	↓

Industry Compatible Mode

This mode stores a word of data as four 8-bit bytes in four frames of a 9-track tape. On a read operation, four frames of 8-bit bytes are read, left-justified, into a word. The remaining four bits of the word are 0 or are copies of the parity bits, depending on the hardware; these bits are not data. On a write operation, the leftmost four 8-bit bytes (i.e., bits 0 through 31) of the word are written in four frames on the tape. The rightmost four bits (i.e., bits 32 through 35) of the word are ignored and are not written on the tape. This mode is compatible with any machine that reads and writes 8-bit bytes.

TRACKS

9	8	7	6	5	4	3	2	1	FRAMES
B0	B1	B2	B3	B4	B5	B6	B7	P	
B8	B9	B10	B11	B12	B13	B14	B15	P	
B16	B17	B18	B19	B20	B21	B22	B23	P	
B24	B25	B26	B27	B28	B29	B30	B31	P	

ANSI ASCII Mode

This mode stores a word of data as five 7-bit bytes in five frames of a 9-track tape. On a read operation, five frames of 7-bit bytes are read, left-justified, into a word. The remaining bits (bits 35) of each frame are ORed together, and the result is placed in bit 35 of the word. On a write operation, the leftmost five 7-bit bytes of the word are written in five frames on the tape. Bit 35 of the word must be zero to conform to ANSI standards. It is written into the high-order bit of the fifth frame, and the remaining high-order bits of the first four frames are 0. This mode is useful when transferring ASCII data from the DECSYSTEM-20 to machines that read 8-bit bytes.

TRACKS

9	8	7	6	5	4	3	2	1	FRAMES
0	B0	B1	B2	B3	B4	B5	B6	P	
0	B7	B8	B9	B10	B11	B12	B13	P	
0	B14	B15	B16	B17	B18	B19	B20	P	
0	B21	B22	B23	B24	B25	B26	B27	P	
B35	B28	B29	B30	B31	B32	B33	B34	P	

SIXBIT Mode

This mode stores a word of data as six 6-bit bytes in six frames of a 7-track tape. This mode is the only supported hardware mode for 7-track tapes.

TRACKS

7	6	5	4	3	2	1	FRAMES
B0	B1	B2	B3	B4	B5	P	
B6	B7	B8	B9	B10	B11	P	
B12	B13	B14	B15	B16	B17	P	
B18	B19	B20	B21	B22	B23	P	
B24	B25	B26	B27	B28	B29	P	
B30	B31	B32	B33	B34	B35	P	

2.4.3 Terminal (TTY:)

Most monitor calls in this group return an error if the device referenced is assigned to another job. However, a process with WHEEL capability enabled can reference a terminal assigned to another job (as controlling terminal or with ASND). The monitor calls pertaining to terminals have no effect, or return default-value information, when used with other devices.

2.4.3.1 **JFN Mode Word** - Each terminal in TOPS-20 is associated with a mode word. This word can be read with the RFMOD call and changed with the SFMOD and STPAR calls. The SFMOD call affects only the modes that are logically program related (i.e., wakeup control, echo mode, and terminal data mode). The STPAR call affects fields that describe device parameters (i.e., mechanical characteristics, page length and width, case conversion, and duplex control). Thus, a program can execute a SFMOD call without affecting previously-established device modes.

The format of the JFN word is shown below.

Bit	Symbol	Changed by	Function
0	TT%OSP	SFMOD	output suppress control (1=ignore output; 0=allow output)
1	TT%MFF	STPAR	has mechanical form feed
2	TT%TAB	STPAR	has mechanical tab
3	TT%LCA	STPAR	has lower case
4-10	TT%LEN	STPAR	page length
11-17	TT%WID	STPAR	page width
18-23	TT%WAK	SFMOD	wakeup control on:
			B18: not used
			B19: not used
	TT%WKF		B20: formatting control character
	TT%WKN		B21: non-formatting control character
	TT%WKP		B22: punctuation character
	TT%WKA		B23: alphanumeric character
24	TT%ECO	SFMOD	echos on
25	TT%ECM	STPAR	echo mode
26	TT%ALK	TLINK	accept links
27	TT%AAD	TLINK	accept advice
28-29	TT%DAM	SFMOD	terminal data mode
	.TTBIN		00: no translation
	.TTASC		01: translate both echo and output
	.TTATO		10: translate output only
	.TTATE		11: translate echo only
30	TT%UOC	STPAR	upper case output control
			0: do not indicate
			1: indicate by 'X
31	TT%LIC	STPAR	lower case input control
			0: no conversion
			1: convert lower to upper
32-33	TT%DUM	STPAR	duplex mode
	.TTFDX		00: Full duplex
	.TTHDX		10: Character half duplex
	.TTLDX		11: Line half duplex
			01: Reserved for DEC
34	TT%PGM	STPAR	output page mode (1=display mode, 0=hardcopy mode)
35	TT%CAR	system	carrier state; on if line is a dataset and the carrier is on.

Bit 0 (TT%OSP) implements the CTRL/O function. If this bit is set, all program output directed to the terminal is discarded. When the bit is off, program output is buffered and sent as usual. The current contents of the output buffer is not cleared when this bit is set; clearing the buffer must be done explicitly (via the CFOBF call) if output is to be stopped immediately.

Bits 1, 2, and 3 (TT%MFF, TT%TAB, and TT%LCA) define several of the mechanical capabilities of the terminal and affect character handling on both input and output. Form feeds and tabs are simulated if the terminal does not have the required mechanical capability or if simulation has been requested by the SFCOC call.

Bits 4-10 (TT%LEN) determine the number of line feeds necessary to simulate a formfeed or the number of lines to fit on the display screen. A 0 value means the declared length of the page is indefinitely large.

Bits 11-17 (TT%WID) determine the point at which the output line must be continued on the next line by inserting a carriage return-line feed. If 0, no line folding occurs.

Bits 18-23 (TT%WAK) define the particular class of characters that, when input from the terminal, will wake up a waiting program. Refer to Section 2.4.3.3 for the definitions of the wakeup classes.

Bit 24 (TT%ECO) defines if echos are to be given. If this bit is off, echoing is turned off. This is useful when the program is accepting a password or is simulating non-standard echoing procedures.

Bit 25 (TT%ECM) defines when the echo will occur. If this bit is off, the echo will occur when the program reads the character. That is, the echo occurs immediately if the program is waiting for input or is deferred if the program is not waiting for input. This is the standard echo mode and produces a correctly-ordered typescript (i.e., program input and output appear in the order in which they occurred). If this bit is on, the echo occurs as soon as the character is typed. Note that this mode may cause editing to appear out of order on the typescript. This occurs because editing is performed as the program reads the character and not necessarily when the echo occurs.

Bits 28-29 (TT%DAM) define the terminal data mode. The four possible data modes are:

00 Binary (.TTBIN), 8-bit input and output. There is no format control or control group translation and no echoing.

01 ASCII (.TTASC), 7-bit input and output, plus parity on for control group output. There is format control as well as simulation and translation of control group for input (echo) and output according to the control words given on the SFCOC JSYS. This is the usual terminal data mode.

10 Disable the translation of echo (.TTATO). In all other respects, same as .TTASC.

11 Disable the translation of output (.TTATE). In all other respects, same as .TTASC.

The last two data modes allow the user to selectively disable the translation of control characters for input or output. When translation is disabled, control characters are always sent. Simulation of formatting control characters is still performed if requested by the control words of the RFCOC or SFCOC JSYS or if the device does not have the required mechanical capability. The translation typically results in some control characters being indicated by graphics instead of being sent as is. For example, disabling the translation of output characters is appropriate for some display terminals when the program must send untranslated control characters to control the display, but requires that the control characters typed by the user be indicated in the usual way.

Bit 30 (TT%UOC) specifies that upper case terminal output is to be indicated by 'X (single quote preceding character that is upper case) on terminals that are not capable of lower case output.

Bit 31 (TT%LIC) specifies that lower case terminal input is to be translated to upper case and that codes 173 througn 176 are to be converted to code 33.

Bits 32-33 (TT%DUM) define the three duplex modes presently available. Full duplex (.TTFDX) requires the system to generate the appropriate echo for each character typed in. Character half duplex (.TTHDX) assumes the terminal will internally echo each character typed but will require an additional echo for formatting characters such as carriage return. Line half duplex (.TTLDX) is similar to character half duplex but does not generate a line feed echo after a carriage return.

Bit 34 (TT%PGM) specifies the output mode. In display mode, the user can create a pause in the output while he reads material that would otherwise quickly disappear off the screen. The output is stopped with the CTRL/S character and started with the CTRL/Q character. Also, output automatically stops whenever a page, as defined by TT%LEN, has been output; output is resumed with CTRL/Q.

Bit 35 (TT%CAR) indicates the carrier state. If the line is a dataset, this bit is on if the carrier is on. If the line is not a dataset, this bit is undefined.

2.4.3.2 **Control Character Output Control** - Each terminal has two control character output control (CCOC) words. Each word consists of 2-bit bytes, one byte for each of the control characters (ASCII codes 0-37). The bytes are interpreted as follows:

00: ignore (send nothing)
01: indicate by ^X (where X is the character)
10: send actual code
11: simulate format action

The specific byte for each character is given in the character set table in Section 2.4.3.3. The CCOC words can be read and manipulated with the RFCOC and SFCOC monitor calls.

2.4.3.3 **Character Set** - The following information describes each character in the TOPS-20 character set that is pertinent to the monitor calls in this group. The wakeup class (refer to TT%WAK in Section 2.4.3.1) is abbreviated as follows:

F formatting control character
C non-formatting control character
P punctuation character
A alphanumeric character

Refer to Section 2.4.3.2 for the explanation of the control character output control (CCOC) words.

ASCII Code	Wakeup Class	CCOC Word(bits)	Character or Control Character	
0	C	1(B0,1)	CTRL/@	null,break
1	C	1(B2,3)	CTRL/A	
2	C	1(B4,5)	CTRL/B	
3	C	1(B6,7)	CTRL/C	
4	C	1(B8,9)	CTRL/D	
5	C	1(B10,11)	CTRL/E	
6	C	1(B12,13)	CTRL/F	
7	C	1(B14,15)	CTRL/G	bell
10	F	1(B16,17)	CTRL/H	backspace
11	F	1(B18,19)	CTRL/I	horizontal tab
12	F	1(B20,21)	CTRL/J	line feed
13	C	1(B22,23)	CTRL/K	vertical tab
14	F	1(B24,25)	CTRL/L	form feed
15	F	1(B26,27)	CTRL/M	carriage return
16	C	1(B28,29)	CTRL/N	
17	C	1(B30,31)	CTRL/O	
20	C	1(B32,33)	CTRL/P	
21	C	1(B34,35)	CTRL/Q	
22	C	2(B0,1)	CTRL/R	
23	C	2(B2,3)	CTRL/S	
24	C	2(B4,5)	CTRL/T	
25	C	2(B6,7)	CTRL/U	
26	C	2(B8,9)	CTRL/V	
27	C	2(B10,11)	CTRL/W	
30	C	2(B12,13)	CTRL/X	
31	C	2(B14,15)	CTRL/Y	
32	C	2(B16,17)	CTRL/Z	
33	all	2(B18,19)	escape (altmode)	
34	C	2(B20,21)	FS	CTRL/backslash
35	C	2(B22,23)	GS	CTRL/right square bracket
36	C	2(B24,25)	RS	CTRL/uparrow
37	F	2(B26,27)	US	CTRL/backarrow
40	P		space	
41	P		!	
42	P		"	
43	P		#	
44	P		$	
45	P		%	
46	P		&	
47	P		'	
50	P		(	
51	P		)	
52	P		*	
53	P		+	
54	P		,	
55	P		-	
56	P		.	
57	P		/	
60-71	A		0-9	
72	P		:	
73	P		;	
74	P		<	
75	P		=	

ASCII Code	Wakeup Class	CCOC Word(bits)	Character or Control Character
76	P		>
77	P		?
100	P		@
101-132	A		upper case letters A-Z
133	P		[
134	P		\
135	P		]
136	P		^
137	P		_
140	P		accent (grave)
141-172	A		lower case letters a-z
173 [1]	P		left brace
174 [1]	P		vertical bar
175 [1]	P		right brace
176 [1]	P		tilde
177	all		delete (rubout)

NOTE

ESC(33) and DELETE(177) are considered to be in all wakeup classes.

---

[1] If the terminal does not have B31(TT%LIC) on in the JFN mode word, codes 173 through 176 are converted to code 33 on input.

2.4.3.4 **Terminal Characteristics Control** - The various types of terminals have different characteristics for output processing, depending on their type and speed. The characteristics that can be associated with terminals are:

1. mechanical form feed and tab

2. lower case

3. padding after carriage return

4. padding after line feed

5. padding after mechanical tab

6. padding after mechanical form feed

7. page width and length

Instead of setting each of these parameters for his line, the user can specify a terminal type number, which causes the appropriate parameters to be set. Refer to the STTYP monitor call. The defined terminal types, along with their characteristics, are listed below.

Number	Terminal	Symbol	Characteristics
0	TTY model 33	.TT33	no mechanical form feed or tab, has upper case only, no padding after carriage return and line feed, padding after tab and form feed, page width 72, page length 66
1	TTY model 35	.TT35	has mechanical form feed and tab, has upper case only, no padding after carriage return and line feed, padding after tab and form feed, page width 72, page length 66
2	TTY model 37	.TT37	no mechanical form feed or tab, lower case, no padding after carriage return and line feed, padding after tab and form feed, page width 72, page length 66
3	TI/EXECUPORT	.TTEXE	no mechanical form feed or tab, lower case, padding after carriage return only page width 80, page length 66
4-7			reserved for customer
8	Default	.TTDEF	no mechanical form feed or tab, lower case, full padding, page width 72, page length 66
9	Ideal	.TTIDL	has mechanical form feed and tab, lower case, no padding, no specified width and length
10	VT05	TTV05	no mechanical form feed, has mechanical tab, has upper case only, no padding after carriage return and tab, padding after line feed and form feed, page width 72, page length 20
11	VT50	.TTV50	no mechanical form feed or tab, has upper case only, no padding, page width 80, page length 12
12	LA30	.TTL30	no mechanical form feed or tab, has upper case only, full padding, page width 80, page length 66
13	GT40	.TTG40	no mechanical form feed or tab, lower case, no padding, page width 80, page length 30

Number	Terminal	Symbol	Characteristics
14	LA36	.TTL36	no mechanical form feed or tab, lower case, no padding, page width 132, page length 66
15	VT52	.TTV52	no mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24

The STTYP monitor call sets the terminal type number for a line, and the GTTYP monitor call obtains the terminal type number.

2.4.3.5 **Terminal Linking** - It is possible to link the output of any line to up to four other lines. The refuse/accept link bit TT%ALK(bit 26) in the JFN mode word controls terminal linking. If the bit is off for a particular terminal, users cannot link to that terminal. Although this bit can be read with the RFMOD monitor call, the bit can only be set with the TLINK call.

Refer to the TLINK monitor call for a description of terminal linking.

2.4.3.6 **Terminal Advising** - It is possible to receive advice from any terminal line in the system. The refuse/accept advice bit TT%AAD (bit 27) in the JFN mode word controls terminal advising. If this bit is off for a particular terminal, users cannot simulate typing on that terminal via the STI monitor call. Although this bit can be read with the RFMOD monitor call, it can only be set with the TLINK call.

Refer to the TLINK monitor call for a description of terminal advising.

2.4.4 Card Reader (CDR:)

The following device-dependent status bits are defined for the card reader. These bits can be obtained with the .MORST function of the MTOPR call.

Bit	Symbol	Meaning
0	MO%COL	Device is on line.
10	MO%FER	Fatal hardware error. This error generates an interrupt on software channel .ICDAE. (Refer to Section 2.5.1.)
12	MO%EOF	Card reader is at end of file.
13	MO%IOP	I/O in progress.
14	MO%SER	Software error (e.g., interrupt character).
15	MO%HE	Hardware error. This error generates an interrupt on software channel .ICDAE.

Bit	Symbol	Meaning
16	MO%OL	Device is off line.
17	MO%FNX	Device is nonexistent.
31	MO%SFL	Output stacker full.
32	MO%HEM	Input hopper empty.
33	MO%SCK	Stack check.
34	MO%PCK	Pick check.
35	MO%RCK	Read check.

The following monitor calls are used when communicating with devices:

ASND	Assigns a device
RELD	Releases a device
SPOOL	Defines and initializes input spooling
LPINI	Loads VFU or translation RAM
DVCHR	Returns device characteristics
GDSTS	Returns the device status
SDSTS	Sets the device status
GDSKC	Returns disk usage
MSTR	Performs structure-dependent functions
MTOPR	Performs device-dependent functions
STDEV	Translates a string to a device designator
DEVST	Translates a device designator to a string
GTTYP	Returns terminal type number
STTYP	Sets terminal type number
ATACH	Attaches controlling terminal to a job
DTACH	Detaches controlling terminal from a job
TLINK	Controls terminal linking
RFMOD	Returns the JFN mode word
SFMOD	Sets program-related fields in the JFN mode word
STPAR	Sets device-related fields in the JFN mode word
RFPOS	Returns current position of the terminal
SFPOS	Sets current position of the terminal
RFCOC	Returns control character output control words
SFCOC	Sets control character output control words
CFIBF	Clears terminal's input buffer
CFOBF	Clears terminal's output buffer
SIBE	Skips if input buffer is empty
SOBE	Skips if output buffer is empty
SOBF	Skips if output buffer is full
DIBE	Dismisses until terminal input buffer is empty
DOBE	Dismisses until terminal output buffer is empty

2.5 SOFTWARE INTERRUPT SYSTEM

The monitor calls in this group are used for controlling the software interrupt system. Note that if the program has an ERJMP or ERCAL after a monitor call that normally causes an interrupt on failure, the ERJMP or ERCAL overrides the interrupt. Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the software interrupt system.

2.5.1 Software Interrupt Channels

Each interrupt is associated with one of 36 software interrupt channels below. The user program can assign channels 0-5 and 23-35 to various conditions, such as terminal interrupts, IPCF interrupts, ENQ/DEQ interrupts, PTY conditions, and terminal buffers becoming empty. The remaining channels are permanently assigned to certain error conditions. Any channel may be used for program-initiated interrupts (IIC call).

Channel	Symbol	Meaning
0-5		Assignable by user program
6	.ICAOV	Arithmetic overflow (includes NODIV)
7	.ICFOV	Arithmetic floating point overflow (includes FXU)
8		Reserved for DEC
9	.ICPOV	Pushdown list (PDL) overflow [1]
10	.ICEOF	End of file condition
11	.ICDAE	Data error file condition [1]
12	.ICQTA	Disk full or quota exceeded when creating a new page [1]
13-14		Reserved for DEC
15	.ICILI	Illegal instruction [1]
16	.ICIRD	Illegal memory read [1]
17	.ICIWR	Illegal memory write [1]
18		Reserved for DEC
19	.ICIFT	Inferior process termination or forced freeze
20	.ICMSE	System resources exhausted [1]
21		Reserved for DEC
22	.ICNXP	Reference to non-existent page
23-35		Assignable by user program

---

[1] These channels are panic channels and cannot be completely deactivated. (Refer to Section 2.5.5.)

2.5.2 Software Interrupt Priority Levels

Each channel is assigned to one of three priority levels. The priority levels are numerically referenced as level 1, 2, or 3 with level 1 being the highest level interrupt. Level 0 is not a legal priority level. If an interrupt request occurs in a process where the level associated with the channel is 0, the system considers the process not prepared to handle the interrupt. The process is then frozen or terminated according to the setting of SC%FRZ (bit 17) in its capabilities word. (Refer to Section 2.6.1.)

2.5.3 Software Interrupt Tables

Before using the software interrupt system, a process must set up the following two tables and declare their addresses with the SIR call.

LEVTAB

A 3-word table, indexed by priority level minus 1. The left half of each word is not used. The right half of each word contains the address in the process' address space in which to store the PC at the time of the interrupt and flags for the associated priority level.

CHNTAB

A 36-word table, indexed by channel number. The left half of each word contains the priority level (1, 2, or 3) for that channel. The right half contains the address of the interrupt routine that will handle interrupts on that channel.

2.5.4 Terminating Conditions

If an interrupt is received on a channel which is activated, but the interrupt cannot be initiated because

1. the interrupt system for the process is not enabled (EIR JSYS) and the channel on which the interrupt occurred is a panic channel,

2. the table addresses have not been defined (SIR call),

3. no priority level has been assigned to the channel (i.e., left half of channel's word in CHNTAB is 0), or

4. the channel has been "reserved" by the superior process (refer to the SIRCM call description),

then the interrupt is considered a process termination condition. In this case the process that was to have received the interrupt is halted or frozen according to the setting of SC%FRZ (bit 17) in its capabilities word, and a process termination interrupt is sent to its superior. The superior process can then execute the RFSTS call to determine the status of the inferior process.

2.5.5 Panic Channels

Panic channels (refer to Section 2.5.1) cannot be completely deactivated by disabling the channel or the entire interrupt system. A software interrupt received on a panic channel that has been deactivated will be considered a process terminating condition. However, panic channels will respond normally to the channel on/off and read channel mask monitor calls.

2.5.6 Terminal Interrupts

There are 36 (decimal) codes used to specify terminal characters or conditions on which interrupts can be initiated. A process can assign a character or condition to any one of the program-assignable interrupt channels with the ATI call. Once the particular code is assigned to a channel and the channel is activated (via AIC), occurrence of the character or condition corresponding to the code causes an interrupt to be generated. The terminal codes, along with their associated conditions, are shown in the table below.

Terminal Code	Symbol	Character or Condition
0	.TICBK	CTRL/@ or break
1	.TICCA	CTRL/A
2	.TICCB	CTRL/B
3	.TICCC	CTRL/C
4	.TICCD	CTRL/D
5	.TICCE	CTRL/E
6	.TICCF	CTRL/F
7	.TICCG	CTRL/G
8	.TICCH	CTRL/H
9	.TICCI	CTRL/I (tab)
10	.TICCJ	CTRL/J (line feed)
11	.TICCK	CTRL/K (vertical tab)
12	.TICCL	CTRL/L (form feed)
13	.TICCM	CTRL/M (carriage return)
14	.TICCN	CTRL/N
15	.TICCO	CTRL/O
16	.TICCP	CTRL/P
17	.TICCQ	CTRL/Q
18	.TICCR	CTRL/R
19	.TICCS	CTRL/S
20	.TICCT	CTRL/T
21	.TICCU	CTRL/U
22	.TICCV	CTRL/V
23	.TICCW	CTRL/W
24	.TICCX	CTRL/X
25	.TICCY	CTRL/Y
26	.TICCZ	CTRL/Z
27	.TICES	escape (altmode)
28	.TICRB	delete (rubout)
29	.TICSP	space
30	.TICRF	dataset carrier off
31	.TICTI	typein
32	.TICTO	typeout
33-35		reserved for DEC

The terminal code .TICRF (30) is used to generate an interrupt when the dataset carrier state changes from on to off. Although any process can enable for this interrupt, only the top-level process in the job is guaranteed to receive it when the carrier state changes. If other processes enable for the interrupt, they can receive the interrupt either when the carrier state changes to off or later when the job is reattached after the detach caused by the carrier-off condition. In general, the occurrence of the change in the dataset carrier state is usable only by the top-level process.

The terminal codes .TICTI (31) and .TICTO (32) are used to generate interrupts on receipt of any character instead of a specific character. The .TICTI code generates an interrupt when the terminal's input buffer becomes nonempty (i.e., when a character is typed and the buffer was empty before the input of the character). The .TICTO code generates an interrupt when the terminal's output buffer becomes nonempty. Note that neither one of these codes generates an interrupt if the buffer is not empty when the character is placed into it. The SIBE and SOBE calls can be used to determine if the buffers are empty.

The frozen or unfrozen state (refer to Section 2.6.2.1) of a process determines if the interrupt is initiated immediately. Terminal interrupts are effectively deactivated when a process is frozen, even though the interrupts are indicated in the process' terminal interrupt word (obtained with the RTIW JSYS). When the process is unfrozen, the terminal interrupts are automatically reactivated.

When an operation is completed that explicitly changes the terminal interrupt word for the job (e.g., a process freeze or unfreeze operation), the interrupt word for the job (and for the terminal line if the job is attached) is set to the inclusive OR (IOR) of all the unfrozen processes in the job. When an interrupt character is received, frozen processes are not considered when searching for a process to interrupt.

The user cannot directly access the actual terminal interrupt word. However, by specifying a process identifier of -5 as an argument to the RTIW or STIW JSYS's, he can read or change the terminal interrupt enable mask. The function of this mask is to allow processes to turn off interrupt codes activated by superior processes. Normally, the mask is -1, thereby enabling all terminal interrupts to be activated. A zero in any position of the mask prevents the corresponding terminal interrupt from being active. However, the fact that a code has been activated will be remembered, and the code will be activated when the mask is changed with a one in the corresponding position. Note that the process must have SC%CTC enabled in its capabilities word (refer to Section 2.6.1) to activate the terminal code for CTRL/C interrupts.

The SCTTY monitor call can be used to change the source of terminal interrupts for a process. Note that the process must have SC%SCT enabled in its capabilities word (refer to Section 2.6.1) to change the source of terminal interrupts.

2.5.6.1 **Terminal Interrupt Modes** - TOPS-20 handles the receipt of a terminal interrupt character in either immediate mode or deferred mode. An interrupt character handled in immediate mode causes the initiation of a software interrupt immediately upon its receipt by the system (i.e., as soon as the user types it). An interrupt character handled in deferred mode is placed in the input stream and initiates a software interrupt only when the program attempts to read it from the input buffer. In either case, the character is not passed to the program. If two occurrences of the same deferred interrupt character are received without any intervening character, the interrupt has an immediate effect. To detect this situation, the system maintains a separate one-character buffer in case the input buffer is otherwise full. The system assumes that interrupts are to be handled immediately unless the process has declared them deferred with the STIW monitor call.

The purpose of deferred mode is to allow interrupt actions to occur in sequence with other actions in the input stream. However, with multiple processes, the deferred interrupt occurs when any process of the job reads the interrupt character. If this process is the one enabled for the interrupt, it will be interrupted before any more characters are passed to the program. If the process to be interrupted is the top process, then the interrupt occurs before more characters are passed to the program, unless another process is also reading from the same source (usually an abnormal condition). If neither of the above situations apply, then the process doing terminal input continues to run and may receive several characters before the interrupt can take effect. This is unavoidable since the process doing input and the process to be interrupted are logically running in parallel.

2.5.7 Dismissing An Interrupt

Once the processing of an interrupt is complete, the user's interrupt routine returns control to the interrupted process via the DEBRK call. When the DEBRK call is executed, the monitor examines the contents of the return PC word to determine where to resume the process. If the PC word has not been changed, the process is restored to its state prior to the interrupt. For example, if the process was dismissed waiting for I/O to complete, it is restored to that state after execution of the DEBRK call. If the PC word has been changed, the process resumes execution at the new PC location.

The process can determine if an interrupt occurred during the execution of monitor code or user code by examining the user/exec mode bit (bit 5) of the return PC word. If the bit is on, the process was executing user code; if the bit is off, the process was executing monitor code (i.e., a JSYS). If the interrupt routine changes the return PC during the processing of an interrupt, the user-mode bit of the new PC word must be on. Note that the process may be executing monitor code but that the address portion of the PC is referencing a location in user code. To return to that user code location (i.e., to interrupt the execution of a monitor call), the process must turn on the user-mode bit.

The monitor calls for controlling the software interrupt system are:

SIR	Sets the interrupt table addresses
RIR	Reads the interrupt table addresses
EIR	Enables the interrupt system
DIR	Disables the interrupt system
CIS	Clears the interrupt system
SKPIR	Skips if the interrupt system is enabled
AIC	Activates interrupt channels
IIC	Initiates interrupts on specific channels in a process
DIC	Deactivates interrupt channels
RCM	Reads activated channel word mask
RWM	Reads waiting channel word mask
SIRCM	Sets inferior reserved channel mask
RIRCM	Reads inferior reserved channel mask
DEBRK	Dismisses current interrupt
ATI	Assigns terminal code to channel
DTI	Deassigns terminal code
STIW	Sets terminal interrupt word
RTIW	Reads terminal interrupt word
GTRPW	Returns trap words
SCTTY	Changes source of terminal interrupts

2.6 PROCESS CAPABILITIES

The TOPS-20 system allows capabilities, such as the ability to examine the monitor and to enable for CTRL/C interrupts, to be given to certain processes. Each capability is separately protected and activated. The capabilities are assigned on a per-process basis, and their status is kept in the process' PSB. The number of capabilities is limited to 36, and two words are used to store the status. For each capability, there is a bit in the first word which is set if the capability is available to the process. If the corresponding bit in the other word is also set, the capability is currently enabled. This allows the user to protect himself against accidental use without actually giving up the capability.

Inferior processes are created by superior processes (via the CFORK monitor call) with either no special capabilities or the capabilities of the creating process. Most capabilities relate to system functions and may be passed from superior to inferior process only if the superior itself has the capability. Some capabilities relate the inferior to the superior process and may be given to an inferior whether or not available in the superior.

2.6.1 Assigned Capabilities

Bit	Symbol	Meaning
		B0-8 Job Capabilities
0	SC%CTC	Process can enable for CTRL/C software interrupts.
1	SC%GTB	Process can examine monitor tables with the GETAB call.
2	SC%MMN	Process can map the running monitor.
3	SC%LOG	Process can execute protected log functions.
4	SC%MPP	Process can map privileged pages of files.
5	SC%SDV	Process can use special devices.
6	SC%SCT	Process can change source of terminal interrupts for other processes.
		B9-17 Capabilities that can be given to an inferior whether or not the superior itself has them. Of these, SC%FRZ (B17) cannot be changed by a process for itself.
9	SC%SUP	Process can manipulate its superior process.
17	SC%FRZ	Unprocessed software interrupts can cause the process to be frozen instead of terminated.
		B18-35 User capabilities
18	SC%WHL	User has wheel privileges.
19	SC%OPR	User has operator privileges.
20	SC%CNF	User has confidential information access.
21	SC%MNT	User has maintenance privileges.

Bit	Symbol	Meaning
22	SC%IPC	User has IPCF privileges.
23	SC%ENQ	User has ENQ/DEQ privileges.
24	SC%NWZ	User has ARPANET wizard privileges.
25	SC%NAS	User nas absolute ARPANET socket privileges.

The capabilities are originally established when the directory is created. (Refer to CRDIR monitor call.)

The capability word can be read with the RPCAP monitor call. Capabilities can be enabled with the EPCAP monitor call.

2.6.2 Processes And Scheduling

These monitor calls deal with establishing and interrogating the process structure of a job. Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the process structure.

2.6.2.1 **Process Freezing** - A superior process can cause one or all of its inferior processes to be frozen. A frozen process is one whose execution is suspended (as soon as it is stoppable from the system's point of view) in such a way that it can be continued at the point it was suspended. A process can be frozen directly or indirectly. A process is directly frozen when its superior makes an explicit request to freeze it. A process is indirectly frozen when its superior is frozen. When a process is directly frozen, all of its inferior processes are indirectly frozen. Therefore, a process can be both directly frozen by its superior process and indirectly frozen if its superior process is subsequently frozen.

The explicit unfreezing of a process clears both its direct freeze and the indirect freeze on all its inferior processes unless an inferior process has a direct freeze. The indirect unfreezing of a process clears only the freeze on that process. This means that an explicit freeze of a process prevents the running of any of its inferior processes, and an explicit unfreezing of a process automatically resumes its inferiors.

The FFORK and RFORK monitor calls are used to freeze and unfreeze processes, respectively. An argument of -4 to these calls directly freezes or resumes all immediately inferior processes, and any processes below the immediately inferior ones are indirectly frozen or resumed. (The freeze and unfreeze operations are never legal on any process that is not inferior to the one executing the monitor call.)

The frozen or unfrozen state of a process can only be changed directly. Thus, monitor calls like SFORK and HFORK change other states of a process but do not affect the frozen state. If the process is frozen and a call is executed that changes one of its states, the process remains frozen and does not begin operating in the changed state until it is resumed. For example, a program can change a frozen process' PC with the SFORK call, but the process will not begin running at the new PC until it is unfrozen. Similarly, the HFORK call can be executed on a frozen process, but the process will not be in the halted state until it is unfrozen. The changed status is always reflected in the information returned by the RFSTS call. In the first example above, RFSTS would return the changed PC, and in the second, it would return the halted code in the status word.

The monitor calls associated with capabilities and processes are:

RPCAP	Returns process capabilities word
EPCAP	Enables process capabilities word
RESET	Resets and initializes current process
CFORK	Creates inferior process
SFORK	Starts process
HFORK	Halts an inferior process
HALTF	Halts a process
DISMS	Dismisses process for specified amount of time
WAIT	Dismisses process until interrupt occurs
WFORK	Waits for process to terminate
KFORK	Kills one or more processes
FFORK	Freezes one or more processes
RFORK	Resumes one or more processes
TFORK	Sets and removes monitor call intercepts
RTFRK	Returns the handle of the process suspended because of a monitor call intercept
UTFRK	Resumes a process suspended because of a monitor call intercept
RFSTS	Returns process' status
SFACS	Sets process' accumulators
RFACS	Returns process' accumulators
PRARG	Sets or returns process argument block
RFRKH	Releases process handles
GFRKS	Gets current process structure
GFRKH	Gets process handle
SPLFK	Splices a process structure
RMAP	Obtains a handle on a page in a process
SPACS	Sets accessibility of page
RPACS	Returns accessibility of page
RWSET	Releases working set
ADBRK	Controls address breaks

2.7 SAVE FILES

TOPS-20 handles two formats of save files: nonsharable and sharable. A nonsharable save file contains sequences of words, whereas a sharable save file contains sequences of pages. The formats of the two types of save files are discussed below.

2.7.1 Format For Nonsharable Save Files

The format of a nonsharable save file is as follows:

```
IOWD   length, address of data

IOWD   length, address of data

XWD    length of entry vector, pointer to first word of
       entry vector
```

2.7.2 Format Of Sharable Save Files

A sharable save file is divided into two main parts: one part contains information about the structure of the file, and the second part contains the data of the file. Currently, page 0 of the save file contains the information about the structure, and pages 1 through n contain the data. (Future releases of TOPS-20 may use more than one page for the structure).

Page 0 of the save file has three distinct sections: the directory section, the entry vector section, and the terminating section. Each of the three sections begins with a word containing its identifier code in the left half and its length in the right half. Each section is described in the paragraphs below.

The directory section is the first of the three sections and describes groups of contiguous pages that have identical access. The format of the directory section is as follows:

```
 0             8 9          17 18                        35

!=======================================================!
!       Identifier code       !       Number of words    !
!            1776             !   (including this word)  !
!                             !   in directory section   !
!=======================================================!
!    Access     !   Page number in file, or 0 if group   !
!     bits      !             of pages is all zero       !
!=======================================================!
!    Repeat     !         Page number in the process     !
!    count      !                                        !
!=======================================================!
/       word pairs to describe each group of pages      /
/              in the process address space             /
!=======================================================!
! Access bits   !          Page number in the file       !
!=======================================================!
! Repeat count  !         Page number in the process     !
!=======================================================!
```

The access bits are determined from the access bits specified by the user on the SSAVE monitor call (refer to the SSAVE call for details). The bits currently defined in the directory section are:

B1 The process page is sharable

B2 The process page is writable

The remaining access bits in the directory section are zero.

The repeat count is the number (minus 1) of consecutive pages in the group described by the word pair. Pages are considered to be in a group when the following three conditions are met:

1. The pages are contiguous.
2. The pages have the same access.
3. The pages either are all zero or are all existent and readable.

A page is considered to be all zero if it is nonexistent or is not readable. A page containing all zeros is considered to be existent. A group of all zero pages is indicated by a file page number of 0.

The word pairs are repeated for each group of pages in the address space.

The entry vector section follows the directory section and points to the entry vector word that is used when the GET monitor call is executed. The format of the entry vector section is as follows:

```
0                          17 18                          35
!=========================================================!
!      Identifier code       !       Number of words      !
!           1775             !    (including this word)   !
!                            !  in entry vector section   !
!=========================================================!
!             Number of words in entry vector             !
!=========================================================!
!                 Address of entry vector                 !
!=========================================================!
```

The data for this section is obtained from the entry vector word. (Refer to Section 2.7.3 for a description of the entry vector.)

The terminating section, called the end section, follows the entry vector section. Its format is as follows:

```
!=========================================================!
!      Identifier code       !                            !
!           1777             !             1              !
!=========================================================!
```

The remaining words in page 0 of the save file are filled with zeros and are ignored by the monitor.

2.7.3 Entry Vector

The entry vector is a block of data that describes entry conditions to be used when the program in the process is executed. The first word of the entry vector contains the program start address, the second word contains the program reenter address, and the third word contains the program version number. (The version number format is: B0-B2 containing the group who last modified the program, B3-B11 containing major version number, B12-B17 containing minor version number, and B18-B35 containing edit number.) Subsequent words in the entry vector can contain data applicable to the particular entry (refer to the GCVEC and GDVEC monitor calls).

Each process has an entry vector word in its process storage block. The format of the entry vector word is:

LH: length of the entry vector (1-777)
RH: address of the first word of the entry vector.

The data for this word is obtained from the entry vector in the save file when a GET monitor call is executed for the file.

The following monitor calls are used in conjunction with save files:

GET	Obtains a saved file
SAVE	Saves a process as nonsharable
SSAVE	Saves a process as sharable
SEVEC	Sets process entry vector
GEVEC	Gets process entry vector
SFRKV	Starts process using its entry vector
SCVEC	Sets compatibility package entry vector
GCVEC	Gets compatibility package entry vector
SDVEC	Sets RMS entry vector
GDVEC	Gets RMS entry vector

2.8 INPUT/OUTPUT CONVERSION

The monitor calls in this group perform input/output conversion. Calls are available to convert in both directions between ASCII text (in core or in a file) and integer numbers, floating point numbers, and TOPS-20 internal dates and times.

2.8.1 Floating Output Format Control

2.8.1.1 Free Format

2.8.1.1 Free Format - The most common format control used with the FLOUT JSYS is free format. This is specified by setting B18-23 (FL%FST) of the format control word to 0. (Refer to Section 2.8.1.2.) Normally, the entire format control word is set to 0; however, certain fields may be specified to force a particular output.

Most numbers greater than or equal to 10^-4 but less than 10^6 (with some exceptions) are output in a typical FORTRAN F format. If the number is an exact integer, it is output with no terminating decimal point unless B6(FL%PNT) is on. If the number is a fraction, it is output as .xxxx with no leading 0's. Nonsignificant trailing zeros in the fraction are never output. A maximum of seven digits will be output if the second field (FL%SND) is not specified. The sign of the number is output only if negative.

If the number is outside the range above, it is output in a typical FORTRAN E format (with some exceptions). The exponent is output as Esxx, where s is the sign output only on negative exponents and xx are the digits of the exponent. The above exceptions about outputting the decimal point and suppressing trailing, nonsignificant zeros apply.

Another free format similar to that above is invoked by specifying a nonzero value for B13-17 (FL%RND) of the format control word. The value in this field specifies the place at which rounding should occur. If this value is 7, the output is the same as if the value were 0 as above. If this value is less than 7, rounding occurs at the specified place, but the output will be as above with a maximum of 7 digits (e.g., 12360 with a rounding specification of 3 will output as 12400). If this value is greater than 7, rounding occurs at the specified position, but more than 7 digits are output. In this case, digits are output until either the rounding specification number is reached or until trailing, nonsignificant zeros are reached.

2.8.1.2 **General Format Control** - The format control word specifies the format for floating point output when free format is not desired. The control word indicates the desired output for the three fields of the number, plus additional control for items such as rounding. The first field of the number is up to the decimal point. The second field is from the decimal point to the exponent. The third field is the exponent.

The format control word is as follows:

Bit	Symbol	Meaning
0-1	FL%SGN	Sign control for first field. The first character position is always used for the minus for negative numbers. For positive numbers, the first character position is defined according to the values below:

Value	Symbol	Meaning
0	.FLDIG	First character is digit.
1	.FLSPC	First character is space.
2	.FLPLS	First character is plus sign.
3	.FLSPA	First character is space.

Bit	Symbol	Meaning
2-3	FL%JUS	Justification control for first field.

Value	Symbol	Meaning
0	.FLLSP	Right justify number using leading spaces.
1	.FLLZR	Right justify number using leading 0's.
2	.FLLAS	Right justify number using leading asterisks.
3	.FLTSP	Left justify number up to decimal point using trailing spaces after third field.

Bit	Symbol	Meaning
4	FL%ONE	Output at least one digit (0 if necessary) in first field.
5	FL%DOL	Prefix the number with a dollar sign ($).
6	FL%PNT	Output a decimal point.
7-8	FL%EXP	Third (exponent) field control.

Value	Symbol	Meaning
0	.FLEXN	No exponent field.
1	.FLEXE	Output E as first character of exponent field.
2	.FLEXD	Output D as first character of exponent field.
3	.FLEXM	Output *10^ as first characters of exponent field.

Bit	Symbol	Meaning
9-10	FL%ESG	Exponent sign control. The first character position is always used for the minus for negative exponents. For positive exponents, the first character position is defined according to the values below:

Value	Symbol	Meaning
0	.FLDGE	First character after exponent prefix is digit.
1	.FLPLE	First character after prefix is plus sign.
2	.FLSPE	First character after prefix is space.
3	.FLDGT	First character after exponent prefix is digit.

Bit	Symbol	Meaning
11	FL%OVL	Use free format on overflow of the first field and expand exponent on overflow of the third field. If this bit is not set, no additional output occurs on column overflow.
13-17	FL%RND	Digit position at which rounding will occur. If field is 0, rounding occurs at the 12th digit. If field is 37, no rounding occurs.
18-23	FL%FST	Number of characters in first field (refer to FL%JUS).
24-29	FL%SND	Number of characters in second field.
30-35	FL%THD	Number of characters in third field.

As an example, to output a number in the format xx.yy, the following bits should be set in AC3 of the FLOUT monitor call.

B4(FL%ONE)	output at least one digit in the first field
B6(FL%PNT)	output a decimal point
B13-B17(FL%RND)	do not round the number
B22	output a maximum of two digits in the first field
B28	output a maximum of two digits in the second field

Examples of numbers output in this format are:

43.86 4.24 0.43

2.8.2 Date And Time Conversion Monitor Calls

As described in Chapter 1, internal date and time is maintained in a 36-bit word. The date is in the left half and is the number of days since November 17, 1858; the time is in the right half and is represented as a fraction of a day. This allows the 36-bit value to be in units of days with a binary point between the left and right halves. The resolution is approximately one-third of a second; that is, the least significant bit represents approximately one-third of a second. The date changes at the transition from 11:59:59 PM to 12:00:00 midnight.

For conversions between local and internal date and time, the time zone in which the installation is located is normally used, with daylight saving applied from 4AM on the next to last Sunday in April to 3:59:59AM on the next to last Sunday in October.

Two monitor calls in this group, IDTIM and ODTIM, convert date and time between text strings (in core or in a file) and internal format. These should satisfy most users. However, there are four more calls, which are subsets of IDTIM and ODTIM. The calls ODTNC, IDTNC, ODCNV, and IDCNV make available separately the conversion between internal format date and time and separate numbers for local year, month, and day, and the conversion between those numbers and text strings. They also provide additional options, which give the caller more control over the conversion performed than IDTIM and ODTIM.

Time zones occur in the calling sequences of the latter four JSYS's. A time zone is represented internally as a number between -12 and 12 decimal, representing the number of hours west of Greenwich. For example, EST is zone 5. Zones -12 and 12 represent the same time but different days because the zones are on opposite sides of the international date line.

The I/O conversion monitor calls are as follows:

NIN	Inputs integer number
NOUT	Outputs integer number
FLIN	Inputs floating-point number
FLOUT	Outputs floating-point number
DFIN	Inputs double-precision, floating-point number
DFOUT	Outputs double-precision, floating-point number
IDTIM	Inputs date and time, converting to internal format
ODTIM	Outputs date and time, converting from internal format to text
IDTNC	Inputs date and time without converting to internal format
ODTNC	Outputs date and time in internal format
IDCNV	Converts from day, month, year to internal date and time
ODCNV	Converts from internal date and time to day, month, year
GTAD	Gets current date and time in internal format

2.9 PRIVILEGED MONITOR CALLS

The following monitor calls are privileged and require the process to have WHEEL or OPERATOR capability enabled:

ALLOC	Allocates a device to a particular job
BOOT	Performs functions required for loading front-end software
CRDIR	Creates or modifies a directory
GTDIR	Returns directory information
DSKOP	Allows hardware address specification of disk transfers
DIAG	Reserves and releases hardware channels
DSKAS	Assigns specific disk addresses
SJPRI	Sets job priority
SPRIW	Sets process priority
HSYS	Specifies system shutdown times
USRIO	Places program in user I/O mode
MSFRK	Starts a process in monitor
PEEK	Reads monitor data
SNOOP	Performs system analysis

SYERR	Records data in the system error file
SMON	Sets various monitor flags
EFACT	Records data in the FACT file
MTALN	Associates magnetic tape drive with logical unit number
TTMSG	Sends a message to a terminal
PMCTL	Controls physical memory
USAGE	Writes entries into the system's accounting data file
UTEST	Tests monitor routines

CHAPTER 3

TOPS-20 MONITOR CALLS

ACCES JSYS 552

Gives a particular type of access to a given directory. The possible types of accesses are:

1. Connecting to a directory on a given structure.

2. Gaining owner and group access rights to directories on a structure without actually connecting to a directory on that structure.

3. Relinquishing owner and group access rights to directories on a structure without disconnecting from a directory on that structure.

The ACCES monitor call can be given for another job if the type of access being requested is for connecting the job (AC%CON) and if the process executing the call has WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1:

B0(AC%CON)	connect the job to the specified directory. After successful completion of the call, the job is connected to and has owner access to the directory, and its default directory becomes this directory.
B1(AC%OWN)	give the job owner access to the specified directory and group access to directories in the same groups as the specified directory. The job's connected directory is unchanged. This function cannot be given for another job or for a files-only directory.
B2(AC%REM)	relinquish the owner access (obtained with the AC%OWN function) to the specified directory and the group access to directories in the same group. The job's connected directory is unchanged. This function cannot be given for another job or for a files-only directory. The settings of B0 and B1 are ignored if B2 is on and the job number given is for the current job.
B18-B35	length of the argument block

AC2: address of the argument block

RETURNS +1: always

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.ACDIR	Pointer to ASCIZ string containing the structure and directory name or a 36-bit directory number.
1	.ACPSW	Pointer to ASCIZ string containing the password of the specified directory. The password is not required if the directory is the job's logged-in directory on a domestic structure and if the directory does not require a password for the owner.
2	.ACJOB	Number of job or -1 for the current job. The process must have WHEEL or OPERATOR capability enabled to give a specific job number other than its own.

The ACCES monitor call is used to implement the CONNECT, ACCESS, and END-ACCESS commands of the TOPS-20 Command Language.

Generates an illegal instruction interrupt on error conditions below.

ACCES ERROR MNEMONICS:

ACESX1: argument block too small

ACESX3: password is required

ACESX4: function not allowed for another job

ACESX5: no function specified for ACCES

ACESX6: directory is not accessed

ACESX7: directory is "files-only" and cannot be accessed

CNDIX1: invalid password

CNDIX5: job is not logged in

STRX01: structure is not mounted

STRX02: insufficient system resources

STRX03: no such directory name

STRX04: ambiguous directory specification

LGINX2: directory is "files-only" and cannot be logged into

CAPX1: WHEEL or OPERATOR capability required

RCDIX2: invalid directory specification

ARGX07: invalid job number

ARGX08: no such job

ADBRK JSYS 570

Controls address breaks. An address break is the suspension of the process when a specified location is referenced in a given manner. This facility is useful when debugging a program.

ACCEPTS IN AC1:		function code in the left half and process handle in the right half
	AC2:	arguments for the
	AC3:	specified function
RETURNS	+1:	always

The available functions are as follows:

Code	Symbol	Meaning
0	.ABSET	Set address break.
1	.ABRED	Read address break.
2	.ABCLR	Clear address break.
3	.ABGAD	Return address of break instruction.

Each function is described in the paragraphs below.

Setting address breaks - .ABSET

This function initializes the address break facility for the specified process. When the process references the location in the manner for which the break has been set, it is suspended. Its superior receives a software interrupt on channel 19 (.ICIFT) if it has enabled for that channel. After processing the interrupt, the superior process can resume the inferior by executing the RFORK monitor call.

Only one address break can be in effect for a process at any one time, and the break affects only the process for which it is set. If another process references the location on which a break is set, it is in no way affected by the break. When an address break is set in a page shared among processes and each process is to be suspended when it references the location, the ADBRK call must be executed for each process.

Breaks cannot be specified for the accumulators.

The .ABSET function requires the following arguments to be given:

AC2: address of location on which to break

AC3: flag word indicating the type of reference on which to break. The following flags are currently defined:

B0(AB%RED)	Break on a read reference.
B1(AB%WRT)	Break on a write reference.
B2(AB%XCT)	Break on an execute (instruction fetch) reference.

Reading address breaks - .ABRED

This function returns the current address break information for the specified process. It returns the following information on a successful return:

AC2: address of location on which a break is set

AC3: flag word indicating the type of reference on which the break will occur. The following flags are currently defined:

B0(AB%RED) Break will occur on a read reference.

B1(AB%WRT) Break will occur on a write reference.

B2(AB%XCT) Break will occur on an execute (instruction fetch) reference.

If no address break has been set for the process, the contents of AC2 and AC3 are zero on return.

Clearing address breaks - .ABCLR

This function removes any address break that was set for the specified process. A program can also remove a break by executing the .ABSET function with AC2 and AC3 containing zero.

Returning the address of the break instruction - .ABGAD

This function returns in AC2 the address of the location on which the process encountered a break. When the location on which the break occurred is in a JSYS routine, the address returned is a monitor PC, not the address of the JSYS. The program can obtain the address of the JSYS by executing an RFSTS monitor call.

Generates an illegal instruction interrupt on error conditions below.

ADBRK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

ABRKX1: address break not available on this system

ARGX02: invalid function

AIC JSYS 131

Activates specific software interrupt channels. (Refer to Section 2.5.)

```
ACCEPTS IN AC1:  process handle

           AC2:  36-bit word
                 Bit n on means activate channel n

RETURNS     +1:  always
```

The DIC monitor call can be used to deactivate specified software interrupt channels.

Generates an illegal instruction interrupt on error conditions below.

AIC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

ALLOC JSYS 520

Allocates a device to a particular job or to the device pool of the monitor's resource allocator. A device under control of the monitor's resource allocator cannot be opened or assigned by any job other than the one to which it is currently allocated. When the allocated device is deassigned, it is returned to the monitor's resource allocator.

```
ACCEPTS IN AC1:  function code (.ALCAL)

           AC2:  device designator

           AC3:  job number, -1, or -2

RETURNS     +1:  failure, error code in AC1

            +2:  success
```

If AC3 contains a job number, then the designated device is allocated to that job.

If AC3 contains -1, then the device is returned to the pool of devices available to all users of the system (i.e., the device is no longer allocated). This is the initial state of all devices.

If AC3 contains -2, then the device is assigned to the monitor resource allocator's pool of devices.

When a non-allocated device (i.e., a device not under control of the resource allocator) is assigned or opened by a job, the device cannot be taken by the resource allocator. However, the resource allocator can allocate the device to the job that currently has it. Thus, when the job releases the device, it will be given to the resource allocator. The allocator receives an IPCF packet when the device is returned to it. The flag word (.IPCFL) of the packet descriptor block has a code of 1(.IPCCC) in the IP%CFC field (bits 30-32). This code indicates the message was sent by the monitor. The first word of the packet data block contains the resource allocator's code .IPCSA. Each following word contains the designator of the returned device. Thus,

```
.IPCFL/<.IPCCC>B32

DATA/.IPCSA
DATA+1/device designator
DATA+2/device designator
```

The ALLOC monitor call requires the process to have WHEEL or OPERATOR capability enabled.

ALLOC ERROR MNEMONICS:

ALCX1: invalid function

ALCX2: WHEEL or OPERATOR capability required

ALCX3: device is not assignable

ALCX4: invalid job number

ALCX5: device already assigned to another job

ALCX6: device assigned to user job, but will be given to allocator when released

ASND JSYS 70

Assigns a device to the caller. The successful return is given if the device is already assigned to the caller.

ACCEPTS IN AC1: device designator

RETURNS +1: failure, error code in AC1

+2: success

The RELD call can be used to release devices assigned to the caller.

ASND ERROR MNEMONICS:

DEVX1: invalid device designator

DEVX2: device already assigned to another job

ASNDX1: device is not assignable

ASNDX2: illegal to assign this device

ASNDX3: no such device

DSMX1: file(s) not closed

ATACH JSYS 116

Detaches the specified job from its controlling terminal (if any) and optionally attaches it to a new controlling terminal. A console-attached entry is appended to the accounting data file.

ACCEPTS IN AC1: B0(AT%CCJ) generate a CTRL/C interrupt to the lowest process in the job that is enabled for a CTRL/C interrupt if the job is currently attached to another terminal. If this bit is not set or if the job is currently not attached to another terminal, the job simply continues running when it is attached.

B1(AT%NAT) do not attach. Prevents both the detaching of the job from its terminal and the attaching of a remote job to the local terminal. Is a no-op unless the remote job has a controlling terminal, in which case the remote job is detached and remains detached. This bit in effect makes ATACH like a remote DTACH.

B2(AT%TRM) attach the given job to the terminal specified in AC4. If this bit is not set, the job is attached to the controlling terminal of the caller.

B18-B35 (AT%JOB) job number of the desired job

AC2: user number under which the job to be attached is logged in. The user number can be obtained with the RCUSR monitor call.

AC3: pointer to the beginning of an ASCIZ password string in the caller's address space

AC4: number of the terminal to be attached to the specified job. This argument is required if B2(AT%TRM) is set.

RETURNS +1: failure, error code in AC1

+2: success. If there is a logged-in job currently attached to the specified terminal, it is detached with primary I/O not redirected. Thus, if a process has primary I/O from the controlling terminal, it will block when it attempts primary I/O and will continue when it is reattached and a character is typed. If there is a job not logged in currently attached to the specified terminal, the job goes away.

It is legal to attach to a job that has a controlling terminal if

1. The job is logged in under the same user name as the job executing the ATACH.

2. The job executing the ATACH supplies the correct password of the job it is attaching to.

3. The job executing the ATACH has WHEEL or OPERATOR capability enabled.

4. The job executing the ATACH has ownership of the job because it created the job (and maintained ownership) with the CRJOB call.

If the controlling terminal is a PTY, a password is not required if either:

1. The owner of the PTY has WHEEL or OPERATOR capability enabled.

2. The specified job is logged in with the same name as the owner of the PTY.

The DTACH monitor call can be used to detach the controlling terminal from the current job.

ATACH ERROR MNEMONICS:

ATACX1: invalid job number

ATACX2: job already attached

ATACX3: incorrect user number

ATACX4: invalid password

ATACX5: this job has no controlling terminal

ATI JSYS 137

Assigns a terminal code to a software interrupt channel. (Refer to Section 2.5.) This call also sets the corresponding bit in the process' terminal interrupt mask. (Refer to the STIW and RTIW monitor calls.)

ACCEPTS IN AC1: terminal code (refer to Section 2.5.6) in the left half and channel number in the right half

RETURNS +1: always

If there is no controlling terminal (i.e., if the job is detached), the assignments are remembered and will be in effect when a terminal becomes attached.

The DTI monitor call can be used to deassign a terminal code.

Generates an illegal instruction interrupt on error conditions below.

ATI ERROR MNEMONICS:

TERMX1: invalid terminal code

ATIX1: invalid software interrupt channel number

ATIX2: control-C capability required

BIN JSYS 50

Inputs the next byte from the specified source. When the byte is read from a file, the file must first be opened, and the size of the byte given, with the OPENF call. When the byte is read from memory, a pointer to the byte is given. This pointer is updated after the call.

ACCEPTS IN AC1: source designator

RETURNS +1: always, with the byte right-justified in AC2

If the end of file is reached, AC2 contains 0 instead of the byte. The program can process this end-of-file condition if an ERJMP or ERCAL is the next instruction following the BIN call.

The BOUT monitor call can be used to output a byte sequentially to a destination.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

BIN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

IOX1: file is not open for reading

IOX4: end of file reached

IOX5: device or data error

BKJFN JSYS 42

Backs up the source designator's pointer by one byte.

ACCEPTS IN AC1: source designator

RETURNS +1: failure, error code in AC1

+2: success, updated string pointer in AC1, if pertinent. (This return actually decrements the pointer.)

The BKJFN call, when referring to a terminal, can be executed only once to back up one character (i.e., the terminal's pointer cannot be backed up twice). The call, when referring to other designators, can be executed more than once in succession.

BKJFN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

BKJFX1: illegal to back up terminal pointer twice

SFPTX2: illegal to reset pointer for this file

SFPTX3: invalid byte number

TTYX01: line is not active

BOOT JSYS 562

Performs basic maintenance and utility functions required for loading and dumping the communications front-end software via the DTE-20. The TOPS-20 system process responsible for performing these functions uses a DIGITAL-supplied protocol to implement them.

The BOOT monitor call requires the process to have WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: function code

AC2: address of argument block

RETURNS +1: always

The available functions and their argument blocks are described below.

Code	Symbol	Meaning
0	.BTROM	Activate the hardware ROM bootstrap in the communications front end. Argument Block 0 .BTDTE DTE-20 number 1 .BTERR Error status flags returned on failure of the call
1	.BTLDS	Load a secondary bootstrap program into the communications front end. The secondary bootstrap, with a maximum size of 256 PDP-11 words, is loaded using the ROM bootstrap. The data to be loaded must be packed as two 16-bit PDP-11 words left justified in each 36-bit DECSYSTEM-20 word. The entire bootstrap program must be loaded at once, and the caller blocks until the transfer is complete. Argument Block 0 .BTDTE DTE-20 number 1 .BTERR Error status flags returned on failure of the call 2 .BTSEC Address of bootstrap program to be loaded
2	.BTLOD	Load the communications front-end memory using the previously loaded secondary or tertiary bootstrap program. The bootstrap program in the front end must abide by the protocol for DTE-20 transfers: the first two bytes of data supplied by the caller must be a count of the remaining number of data bytes. Argument Block 0 .BTDTE DTE-20 number 1 .BTERR Error status flags returned on failure of the call 2 Not used and must be zero. 3 .BTFLG User-supplied flag word B0(BT%BEL) Send a doorbell to the front end to indicate when the setup is complete and the transfer can begin. 4 .BTCNT Number of bytes to transfer 5 .BTLPT Pointer to data to be loaded

3	.BTDMP	Dump the communications front-end memory using the ROM bootstrap program. The caller must activate the ROM bootstrap (function .BTROM) before dumping memory. Subsequent .BTDMP functions to dump memory start where the previous dump terminated unless the ROM bootstrap is activated again by a .BTROM function. The caller blocks until the transfer is complete.

Argument Block

0	.BTDTE	DTE-20 number
1	.BTERR	Error status flags returned on failure of the call
2		Not used and must be zero.
3	.BTFLG	User-supplied flag word. This word is not used and must be zero.
4	.BTCNT	Number of bytes to transfer
5	.BTDPT	Pointer to where the data is to be dumped in the DECSYSTEM-20

4	.BTIPR	Initialize the protocol to be used with this communications front end. After successful execution of this function, the DECSYSTEM-20 will process interrupts from the given DTE-20.

Argument Block

0	.BTDTE	DTE-20 number
1	.BTPRV	Version number of the protocol to be used

5	.BTTPR	Stop the protocol currently running on this communications front end. After successful execution of this function, the DECSYSTEM-20 will ignore interrupts from the given DTE-20.

Argument Block

0	.BTDTE	DTE-20 number

6	.BTSTS	Return the status type of the protocol running on this communications front end.

Argument Block

0	.BTDTE	DTE-20 number
1	.BTCOD	Returned protocol version type (e.g., .VN20F for the RSX20F protocol). If no protocol is running, this word contains -1.

7	.BTBEL	Block until a signal (doorbell) to the DECSYSTEM-20 is initiated by the communications front end. This function is used to synchronize the caller with the bootstrap program in the front end. Argument Block 0 .BTDTE DTE-20 number
10	.BTRMP	Read data from the communications front end using the previously loaded secondary or tertiary bootstrap program. The bootstrap program must abide by the protocol for DTE-20 transfers. The first two bytes of data will be interpreted as a count of the remaining number of bytes of data. Argument Block

0	.BTDTE	DTE-20 number
1	.BTERR	Error status flags returned on failure of the call
2		Not used and must be zero.
3	.BTFLG	User-supplied flag word B0(BT%BEL) Send a signal (doorbell) to the DECSYSTEM-20 to indicate the transfer is finished.
4	.BTCNT	Maximum number of bytes to transfer. After successful execution of this function, this word is updated to reflect the actual number of bytes transferred.
5	.BTMPT	Pointer to where data is to be placed

The error status flag returned in word .BTERR on failure of a BOOT call are front-end reload status bits recorded in the SYSERR error file. (Refer to the DECSYSTEM-20 Error Detection, Recovery, and Reporting Reference Manual for an explanation of these status bits.)

Generates an illegal instruction interrupt on error conditions below.

BOOT ERROR MNEMONICS:

BOTX01: invalid DTE-20 number

BOTX02: invalid byte size

BOTX03: invalid protocol version number

BOTX04: byte count is not positive

BOTX05: protocol initialization failed

CAPX1: WHEEL or OPERATOR capability required

ARGX02: invalid function

BOUT JSYS 51

Outputs a byte sequentially to the specified destination. When the byte is written to a file, the file must first be opened, and the size of the byte given, with the OPENF call. When the byte is written to memory, a pointer to the location in which to write the byte is given in AC1. This pointer is updated after the call.

ACCEPTS IN AC1: destination designator

AC2: the byte to be output, right-justified

RETURNS +1: always

The BIN monitor call can be used to input a byte sequentially from a source.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

BOUT ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

IOX2: file is not open for writing

IOX5: device or data error

IOX6: illegal to write beyond absolute end of file

IOX11: quota exceeded or disk full

CACCT JSYS 4

Changes the account for the current job.

ACCEPTS IN AC1: account number in bits 3-35 if bits 0-2 are 5. Otherwise contains a pointer to the new account string in the address space of caller. If a null byte is not seen, the string is terminated after 39 characters are processed.

RETURNS +1: failure, error code in AC1

+2: success, updated string pointer in AC1

The CACCT call sets the current account for the job to the specified account. Subsequent session charges will be to this new account. This call also validates the account given if the account validation facility is enabled. (Refer to the .SFAVR function of the SMON/TMON monitor call.)

The GACCT monitor call can be used to return the account for the current job.

CACCT ERROR MNEMONICS:

CACTX1: invalid account identifier

CACTX2: job is not logged in

CFIBF JSYS 100

Clears the designated file input buffer.

ACCEPTS IN AC1: source designator

RETURNS +1: always

Is a no-op if the source designator is not associated with a terminal.

The CFOBF monitor call can be used to clear a designated file output buffer.

Generates an illegal instruction interrupt on error conditions below.

CFIBF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

CFOBF JSYS 101

Clears the designated file output buffer.

ACCEPTS IN AC1: destination designator

RETURNS +1: always

Is a no-op if the destination designator is not associated with a terminal.

The CFIBF call can be used to clear a designated file input buffer.

Generates an illegal instruction interrupt on error conditions below.

CFOBF ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX3:	JFN is not assigned
DESX5:	file is not open
DEVX2:	device already assigned to another job
TTYX01:	line is not active

CFORK JSYS 152

Creates a process inferior to this process. (Refer to Section 2.6.2.)

ACCEPTS IN AC1:

B0(CR%MAP)	make the inferior process' map the same as the current process' map by means of indirect pointers. If this bit is not on, the inferior process' map is set to 0.
B1(CR%CAP)	make the inferior process' capabilities the same as the current process'. If this bit is not on, the inferior process has no special capabilities.
B3(CR%ACS)	set the inferior process' ACs from the block whose address is in AC2. If this bit is not on, the inferior process' ACs are set to 0.
B4(CR%ST)	set the PC of the inferior process to the value in the right half of AC1 and start the process. If this bit is not on, the inferior process is not started, and the right half of AC1 is ignored.
B18-B35 (CR%PCV)	PC value for the inferior process if CR%ST is on.

AC2: address of 20(octal) word block (optional). This block contains the AC values for the inferior process. (Refer to bit CR%ACS above.)

RETURNS +1: failure, error code in AC1

+2: success, relative process handle in AC1

The inferior process receives the same primary input and output JFN's as the current process. However, the primary input and/or output files may be changed with the SPJFN monitor call.

The CR%MAP argument in AC1 allows the inferior to see the same address space as that of the superior. The inferior process will have read and write access to the superior's address space. The pages are shared, and changes made by one process will be seen by the other.

The KFORK monitor call can be used to kill one or more processes.

CFORK ERROR MNEMONICS:

FRKHX6: all relative process handles in use

CFRKX3: insufficient system resources

CHFDB JSYS 64

Changes a word in the file descriptor block (FDB) for the specified file. (Refer to Section 2.2.8 for the format of this block and the list of words that can be changed with the CHFDB call.)

The CHFDB monitor call cannot be used to change the name string of either the author of the file or the user who last wrote the file (words .FBAUT and .FBLWR of the FDB). The SFUST call must be used to change these words. In addition, the SACTF call is used to change the account designator of the file (word .FBACT).

ACCEPTS IN AC1:

B0(CF%NUD)	do not wait for the disk copy of the directory to be updated. The specified changes are made to the directory in memory and are written to the disk as a part of the normal monitor disk updating procedure. (See below for more information.)
B9-B17 (CF%DSP)	index into FDB indicating word to be changed
B18-B35 (CF%JFN)	JFN

AC2: mask indicating bits to be changed

AC3: new values for changed bits. These values must be given in the bit positions corresponding to the mask given in AC2.

RETURNS +1: always

Because each CHFDB call changes only one word in the FDB, several calls must be executed to change several words. Each call causes disk I/O, and to keep this I/O to a minimum, the program should set bit CF%NUD on each call. The setting of this bit on each call permits the program to run faster by allowing several changes to be made to the FDB with minimum disk I/O. To ensure that all the changes have been written to the disk, the program can issue the last CHFDB call with bit CF%NUD off. Also, if the program requires the FDB on the disk to be updated after each call, it should execute each CHFDB call with bit CF%NUD off.

The GTFDB monitor call can be used to return the file descriptor block for a specified file. The GFUST monitor call can be used to return the name strings of the author of the file and the user who last wrote the file. The GACTF call can be used to obtain the account designator of the file.

Generates an illegal instruction interrupt on error conditions below.

CHFDB ERROR MNEMONICS:

CFDBX1:	invalid displacement
CFDBX2:	illegal to change specified bits
CFDBX3:	write or owner access required
CFDBX4:	invalid value for specified bits
DESX1:	invalid source/destination designator
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX7:	JFN cannot refer to output wildcard designators

CHKAC JSYS 521

Checks if a user is allowed access to files in a given directory. This monitor call is used to determine if the user will be able to access files having a given protection code if the user is logged in with the given capabilities and connected to the directory.

ACCEPTS IN AC1: length of the argument block in the right half. If B0(CK%JFN) is on, word .CKAUD of the argument block contains a JFN.

AC2: address of argument block

RETURNS +1: failure, error code in AC1

+2: success, access check is completed, with AC1 containing -1 if access is allowed or 0 if access is not allowed.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.CKAAC	Code of desired access to files.
1	.CKALD	Pointer to user name string, or 36-bit user number of user whose access is being checked.
2	.CKACD	Pointer to directory name string, or 36-bit directory number to which user whose access is being checked is connected.
3	.CKAEC	Enabled capabilities of user whose access is being checked. (Refer to Section 2.6.1.)
4	.CKAUD	Pointer to directory name string, or 36-bit directory number of the directory containing the files being accessed. If B0(CK%JFN) of AC1 is on, this word contains a JFN for the file being accessed.
5	.CKAPR	Protection of the files being accessed. (Refer to Section 2.2.6.) This word is not required if a JFN is supplied in word .CKAUD.

The codes of the desired accesses are as follows:

0	.CKARD	read existing files
1	.CKAWR	write existing files
2	.CKAEX	execute existing files
3	.CKAAP	append to existing files
4	.CKADL	obtain directory listing of existing files
6	.CKADR	read the directory
10	.CKACN	connect to the directory
11	.CKACF	create files in the directory

CHKAC ERROR MNEMONICS:

CKAX1: argument block too small

CKAX2: invalid directory number

CKAX3: invalid access code

CKAX4: file is not on disk

CIS JSYS 141

Clears the software interrupt system for the current process. Clears all interrupts in progress and all waiting interrupts.

RETURNS +1: always

CLOSF JSYS 22

Closes a specific file or all files.

ACCEPTS IN AC1:	B0(CO%NRJ)	do not release the JFN
	B6(CZ%ABT)	abort any output operations currently being done. Close the file but do not perform any cleanup operations normally associated with closing a file (e.g., do not output remaining buffers or write tape marks if output to a magnetic tape is aborted). If output to a new disk file that has not been closed (i.e., is nonexistent) is aborted, the file is closed and then expunged.
	B7(CZ%NUD)	do not update the copy of the directory on the disk. (Refer to CF%NUD of the CHFDB call description for further information.)
	B18-B35 (CO%JFN)	JFN of the file being closed

RETURNS	+1:	failure, error code in AC1
	+2:	success

If the contents of AC1 is -1, all files (and all JFN's) at or below this process (with the exception of the primary I/O files and files that cannot be closed by this process) are closed. This action is identical to that taken on a CLZFF call with AC1 containing the process handle .FHSLF (400000).

The OPENF monitor call can be used to open a specific file.

CLOSF ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
CLSX1:	file is not open
CLSX2:	file cannot be closed by this process
CLSX3:	file still mapped
CLSX4:	device still active
ENQX20:	locked JFN cannot be closed
IOX11:	quota exceeded or disk full

All output errors may occur.

CLZFF JSYS 34

Closes process' files. Closes all files and/or releases all JFN's at and/or below a specified process.

ACCEPTS IN AC1:

B0(CZ%NIF)	do not close files of inferior processes
B1(CZ%NSF)	do not close files of this process
B2(CZ%NRJ)	do not release JFNs
B3(CZ%NCL)	do not close any files; only release non-open JFNs
B4(CZ%UNR)	unrestrict files opened with restricted access for specified process. The specified process must be the same as or inferior to the process executing the call.
B5(CZ%ARJ)	wait until file can be closed, then close it, and release JFNs
B6(CZ%ABT)	abort any output operations currently being done. Close the file but do not perform any cleanup operations normally associated with closing a file (e.g., do not output remaining buffers or write tape marks if output to a magnetic tape is aborted). If output to a new disk file that has not been closed (i.e., is nonexistent) is aborted, the file is closed and then expunged.
B7(CZ%NUD)	do not update the copy of the directory on the disk. (Refer to CF%NUD of the CHFDB call description for further information.)
B18-B35 (CZ%PRH)	process handle

RETURNS +1: always. No action is taken if the call is in any way illegal.

If AC1 contains only the process handle .FHSLF, the action is identical to that taken on a CLOSF call with AC1 containing -1.

Generates an illegal instruction interrupt on error conditions below.

CLZFF ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

IOX11: quota exceeded or disk full

COMND JSYS 544

Parses one field of a command that is either typed by a user or contained in a file. When this monitor call is used to read a command from a terminal, it provides the following features:

1. Allows the input of a command (including the guide words) to be given in abbreviated, recognition (ESC and CTRL/F), and/or full input mode.

2. Allows the user to edit his input with the DELETE, CTRL/U, CTRL/W, and CTRL/R editing keys.

3. Allows fields of the command to be defaulted if an ESC or CTRL/F is typed at the beginning of any field or if a field is omitted entirely.

4. Allows a help message to be given if a question mark (?) is typed at the beginning of any field.

5. Allows input of an indirect file (@file) that contains the fields for all or the remainder of the command.

6. Allows a recall of the correct portion of the last command (i.e., up to the beginning of the field where an error was detected) if the next command line begins with CTRL/H. The correct portion of the command is retyped, and the user can then continue typing from that point.

7. Allows input of a line to be continued onto the next line if the user types a hyphen (-) immediately preceding a carriage return. (The carriage return is invisible to the program executing the COMND call, although it is stored in the text buffer.) The hyphen can be typed by the user while he is typing a comment. The comment is then continued onto the next line.

The COMND call allows the command line that is input to contain a comment if the comment is preceded by either an exclamation point or a semicolon and the previous field has been terminated. When the COMND call inputs an exclamation point after a field that has been terminated, it ignores all text on the remainder of the line or up to the next exclamation point. When the COMND call inputs a semicolon after a field that has been terminated, it ignores all text on the remainder of the line.

When an indirect file is given on the command line, it can be given at the beginning of any field. However, it must be the last item typed on the line, and its contents must complete the current command. The user must terminate his input of the indirect file (after any recognition is performed) with a carriage return. If he does not terminate his input, the message ?INDIRECT FILE NOT CONFIRMED is output. Also, if the user types a question mark (instead of the file specification of the indirect file) after he types the @ character, the message FILESPEC OF INDIRECT FILE is output. The indirect file itself should not contain an ESC or carriage return; if these characters are included, they will be treated as spaces. The contents of the indirect file are placed in the text buffer but are not typed on the user's terminal.

As the user types his command, the characters are placed in a **command text buffer**. This buffer can also include the command line prompt, if any. Several pointers and counts reflect the current state of the parsing of the command. These pointers and counts are as follows:

1. Pointer to the beginning of the prompting-text buffer (.CMRTY). This pointer is also called the CTRL/R buffer pointer since it indicates the initial part of the text that will be output on a CTRL/R. (The remainder of the text output on a CTRL/R is what the user had typed before he typed CTRL/R.) The buffer containing the prompt need not be contiguous with the buffer containing the remainder of the command line. Typically this pointer is to a string in the literals area.

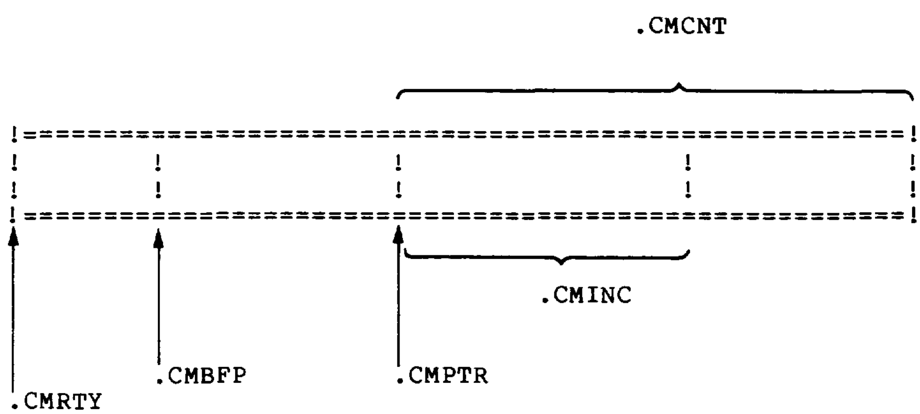
2. Pointer to the beginning of the user's input (.CMBFP). This is the limit back to which the user can edit.

3. Pointer to the beginning of the next field to be parsed (.CMPTR).

4. Count of the space remaining in the text input buffer (.CMCNT).

5. Count of the number of characters in the buffer that have not yet been parsed (.CMINC).

The illustration below is a logical arrangement of the pointers and counts. Remember that the prompting-text buffer does not have to be adjacent to the text buffer.



These pointers and other information are contained in a **command state block**, whose address is given as an argument to the COMND monitor call. The .CMINI function initializes these pointers.

Parsing of a command is performed field by field and by default begins when the user types a carriage return, ESC, CTRL/F, or question mark. These characters are called action characters because they cause the system to act on the command as typed so far. A field can also be terminated with a space, tab, slash, comma, or any other nonalphanumeric character. Normally, the parsing does not begin, and the COMND call does not return control to the program, until an action character is typed. However, if B8(CM%WKF) is on in word .CMFLG when the COMND call is executed, parsing begins after each field is terminated.

The command is parsed by repeated COMND calls. Each call specifies the type of field expected to be parsed by supplying an appropriate function code and any data needed for the function. This information is given in a **function descriptor block.** On successful completion of each call, the current text pointers and the counts are updated in the command state block, and any data obtained for the field is returned.

The program executing the COMND call should not reset the pointers in the command state block after it completes the parsing of each command. It should set up the state block once at the beginning and then use the .CMINI function when it begins parsing each line of a command. This is true because the .CMINI function implements the CTRL/H error recovery feature in addition to initializing the pointers in the state block and printing the prompt for the line. If the program resets the pointers, the CTRL/H feature is not possible because the pointers from the previous command are not available. When a CTRL/H is input, the .CMINI function allows error recovery from the last command only if both (1) the pointer to the beginning of the user's input (.CMBFP) is not equal to the pointer to the beginning of the next field to be parsed (.CMPTR) and (2) the last character parsed in the previous command was not an end-of-line character.

The design of the COMND call allows the user to delete his typed input with the DELETE, CTRL/W, and CTRL/U keys without regard to field boundaries. When the user deletes into a field that has already been parsed, the COMND call returns to the program with B3(CM%RPT) set in word .CMFLG. This return informs the program to forget the current state of the command and to reparse from the beginning of the line. Because the complete line as typed and corrected by the user is in the text buffer, the parse can be repeated and will yield the same result up to the point of the change.

The calling sequence to the COMND call is as follows:

ACCEPTS IN AC1: address of the command state block

AC2: address of the first alternative function descriptor block

RETURNS +1: always (unless a reparse is needed and the right half of .CMFLG is nonzero), with

- AC1 containing flags in the left half, and the address of the command state block in the right half. The flags are copied from word .CMFLG in the command state block.
- AC2 containing either the data obtained for the field or an error code if the field could not be parsed (CM%NOP is on).
- AC3 containing in the left half the address of the function descriptor block given in the call, and in the right half the address of the function descriptor block actually used (i.e., the one that matched the input).

The format of the command state block is shown below.

```
         0                        17 18                        35
        !=======================================================!
 .CMFLG !        Flag Bits         ! Reparse Dispatch Address   !
        !-------------------------------------------------------!
 .CMIOJ !        Input JFN         !         Output JFN         !
        !-------------------------------------------------------!
 .CMRTY !              Pointer to CTRL/R Buffer                 !
        !-------------------------------------------------------!
 .CMBFP !           Pointer to Start of Text Buffer             !
        !-------------------------------------------------------!
 .CMPTR !          Pointer to Next Input To Be Parsed           !
        !-------------------------------------------------------!
 .CMCNT !             Count of Space Left in Buffer             !
        !-------------------------------------------------------!
 .CMINC !          Count of Characters Left in Buffer           !
        !-------------------------------------------------------!
 .CMABP !                Pointer to Atom Buffer                 !
        !-------------------------------------------------------!
 .CMABC !                  Size of Atom Buffer                  !
        !-------------------------------------------------------!
 .CMGJB !           Address of GTJFN Argument Block             !
        !=======================================================!
```

Command State Block

Word	Symbol	Meaning
0	.CMFLG	Flag bits in the left half, and the reparse dispatch address in the right half. Some flag bits can be set by the program executing the COMND call; others can be set by the COMND call after its execution. The bits that can be set by the program are described following the Command State Block description. The bits that can be set by COMND are described following the Function Descriptor Block description. The reparse dispatch address is the location to which control is automatically transferred when a reparse of the command is needed because the user edited past the current pointer (i.e., the user edited characters that were already parsed). If this field is zero, the COMND call sets B3(CM%RPT) in the left half of this word and gives the +1 return when a reparse is needed. The program must then test CM%RPT and, if on, must reenter the code that parses the first field of the command. When the reparse dispatch address is given, control is transferred automatically to that address. The code at the reparse dispatch address should initialize the program's state to what it was after the last .CMINI function. This initialization should include resetting the stack pointer, closing and releasing any JFNs acquired since the last .CMINI function, and transferring control to the code immediately following the last .CMINI function call.

1	.CMIOJ	Input JFN in the left half, and output JFN in the right half. These designators identify the source for the input of the command and the destination for the output of the typescript. These designators are usually .PRIIN (for input) and .PRIOU (for output).
2	.CMRTY	Pointer to the beginning of the prompting-text input buffer (also called the CTRL/R buffer).
3	.CMBFP	Pointer to the beginning of the user's input. The user cannot edit back past this pointer.
4	.CMPTR	Pointer to the beginning of the next field to be parsed.
5	.CMCNT	Count of the space remaining in the buffer after the .CMPTR pointer.
6	.CMINC	Count of the number of unparsed characters in the buffer after the .CMPTR pointer.
7	.CMABP	Pointer to the atom buffer, a temporary storage buffer that contains the last field parsed by the COMND call. The terminator of the field is not placed in this buffer. The atom buffer is terminated with a null.
10	.CMABC	Count of the number of characters in the atom buffer. This count should be at least as large as the largest field expected to be parsed.
11	.CMGJB	Address of a GTJFN argument block. This block must be at least 16(octal) words long and must be writable. If a longer GTJFN block is being reserved, the count in the right half of word .GJF2 of the GTJFN argument block must be greater than four. This block is usually filled in by the COMND call with arguments for the GTJFN call if the specified function is requesting a JFN (i.e., functions .CMIFI, .CMOFI, and .CMFIL). The user should store data in this block on the .CMFIL function only.

The flag bits that can be supplied in the left half of word .CMFLG in the Command State Block are described below. These bits apply to the parsing of the entire command and are preserved by COMND after execution.

Bits Supplied on COMND Call

Bit	Symbol	Meaning
6	CM%RAI	Convert lowercase input to uppercase.
7	CM%XIF	Do not recognize the @ character as designating an indirect file; instead consider the character as ordinary punctuation. A program sets this bit to prevent the input of an indirect file.

8	CM%WKF	Begin parsing after each field is terminated instead of only after an action character (carriage return, ESC, CTRL/F, question mark) is typed. For example, a program sets this bit if it must change terminal characteristics (e.g., it must turn off echoing because a password may be input) in the middle of a command. However, use of this bit is not recommended because terminal wakeup occurs after each field is terminated, thereby increasing system overhead. The recommended method of changing terminal characteristics within a command is to input the field requiring the special characteristic on the next line with its own prompt. For example, if a program is accepting a password, it should turn off echoing after the .CMCFM function of the main command and perform the .CMINI function to type the prompt requesting a password on the next line.

The format of the function descriptor block is shown below.

```
       0          8 9          17 18                         35
       !=======================================================!
       !  function   !  function   ! address of next function  !
 .CMFNP!    code     !    flags    !     descriptor block      !
       !-------------------------------------------------------!
 .CMDAT!               Data for specific function              !
       !-------------------------------------------------------!
 .CMHLP!             Pointer to help text for field            !
       !-------------------------------------------------------!
 .CMDEF!           Pointer to default string for field         !
       !=======================================================!
```

Function Descriptor Block

Word	Symbol	Meaning
0	.CMFNP	Function code and pointer to next function descriptor block. B0-B8(CM%FNC) Function code B9-B17(CM%FFL) Function-specific flags B18-B35(CM%LST) Address of the next function descriptor block
1	.CMDAT	Data for the specific function, if any.
2	.CMHLP	Pointer to the help text for this field. This word can be zero if the program is not supplying its own help text.
3	.CMDEF	Pointer to the default string for this field. This word can be zero if the program is not supplying its own default string.

The individual words in the function descriptor block are described in the following paragraphs.

Words .CMFNP and .CMDAT of the function descriptor block

Word .CMFNP contains the function code for the expected field to be parsed, and word .CMDAT contains any additional data needed for that function. The function codes, along with any required data for the functions, are described below.

Code	Symbol	Meaning
0	.CMKEY	Parse a keyword, such as a command name. Word .CMDAT contains the address of a keyword symbol table in the format described in the TBLUK monitor call description (i.e., alphabetical). The data bits that can be defined in the right half of the first word of the argument pointed to by the table entries (when B0-B6 of the first word are off and B7(CM%FW) is on) are as follows:

B35(CM%INV)	Suppress this keyword in the list output on a ?. The program can set this bit to include entries in the table that should be invisible because they are not preferred keywords. For example, this bit can be set to allow the keyword LIST to be valid, even though the preferred keyword may be PRINT. The LIST keyword would not be listed in the output given on a ?. This bit is also used in conjunction with the CM%ABR bit to suppress an abbreviation in the output given on a ?.
B34(CM%NOR)	Do not recognize this keyword even if an exact match is typed by the user and suppress its listing in the list output on a ?. (Refer to the TBLUK call description for more information on using this bit.)
B33(CM%ABR)	Consider this keyword a valid abbreviation for another entry in the table. The right half of this table entry points to the keyword for which this is an abbreviation. The program can set this bit to include entries in the table that are less than the minimum unique abbreviation. For example, this bit can be set to include the entry ST (for START) in the table. If the user then types ST as a keyword, it will be accepted as a valid abbreviation even though there may be other keywords beginning with ST. To suppress the output of this abbreviation in the list typed on a ?, the program must also set the CM%INV bit.

On a successful return, AC2 contains the address of the table entry where the keyword was found.

1	.CMNUM	Parse a number. Word .CMDAT contains the radix (from 2 to 10) of the number. On a successful return, AC2 contains the number.
2	.CMNOI	Parse a guide word string, but do not return an error if no guide word is input. An error is returned only if a guide word is input that does not match the one expected by the COMND call. A guide word field must be delimited by parentheses. Word .CMDAT contains a pointer to an ASCIZ string. This string does not contain the parentheses of the guide word. Guide words are output if the user terminated the previous field with ESC. Guide words are not output, nor can they be input, if the user has caused parsing into the next field.
3	.CMSWI	Parse a switch. A switch field must begin with a slash and can be terminated with a colon in addition to any of the legal terminators. Word .CMDAT contains the address of a switch keyword symbol table. (Refer to the TBLUK monitor call description for the format of the table.) The entries in the table do not contain the slash of the switch keywords; however, they should end with a colon if the switch requires a value. The data bits CM%INV, CM%NOR, and CM%ABR defined for the .CMKEY function can also be set on this function. On a successful return, AC2 contains the address of the table entry where the switch keyword was found.
4	.CMIFI	Parse an input file specification. This function causes the COMND call to execute a GTJFN call to attempt to parse the specification for an existing file, using no default fields. The .CMGJB address (word 11 in the command state block) must be supplied, but no data should be stored in the block. (Data stored in the block will be overwritten by this COMND call.) On a successful return, AC2 contains the JFN assigned.
5	.CMOFI	Parse an output file specification. This function causes the COMND call to execute a GTJFN call to attempt to parse the specification for either a new or an existing file. The default generation number is the generation number of the existing file plus 1. The .CMGJB address must be supplied, but no data should be stored in the block. On a successful return, AC2 contains the JFN assigned.
6	.CMFIL	Parse a general (arbitrary) file specification. This function causes the COMND call to execute a GTJFN to attempt to parse the specification for the file. The .CMGJB address must be supplied, but no data should be stored in words .GJSRC, .GJCPP, .GJCPC, and .GJRTY of the GTJFN block. Also, the COMND call sets the following flag bits in the GTJFN block: GJ%XTN, G1%RND, G1%RBF, G1%RCM, and G1%RIE. (Refer to the long-form GTJFN

6	.CMFIL (Cont.)	call description for an explanation of these words and flag bits.) The program can set any other words and flag bits in the GTJFN block it supplies. On a successful return, AC2 contains the JFN assigned.
7	.CMFLD	Parse an arbitrary field. This function is useful for fields not normally handled by the COMND call. The input, as delimited by the first nonalphanumeric character, is copied into the atom buffer; the delimiter is not copied. No verification is performed nor is any standard help message available. (See below.)
10	.CMCFM	Confirm. This function waits for the user to confirm the command with a carriage return and should be used at the end of parsing a command line.
11	.CMDIR	Parse a directory name. Login and files-only directories are allowed. Word .CMDAT contains data bits for this function. The currently defined bit is as follows: B0(CM%DWC) Allow wildcard characters to be typed in a directory name. On a successful return, AC2 contains the 36-bit directory number.
12	.CMUSR	Parse a user name. Only login directories are allowed. On a successful return, AC2 contains the 36-bit user number.
13	.CMCMA	Comma. Sets B1(CM%NOP-no parse) in word .CMFLG of the command state block and returns if a comma is not the next item in the input. Blanks can appear on either side of the comma. This function is useful for parsing a list of arguments.
14	.CMINI	Initialize the command line (e.g., set up internal monitor pointers and type the prompt). This function should be used at the beginning of parsing a command line but not when reparsing a line.
15	.CMFLT	Parse a floating-point number. On a successful return, AC2 contains the floating-point number.
16	.CMDEV	Parse a device name. On a successful return, AC2 contains the device designator.
17	.CMTXT	Parse the input text up to the next carriage return, place the text in the atom buffer, and return. If an ESC or CTRL/F is typed, it causes the terminal bell to ring (because recognition is not available with this function) and is otherwise ignored. If a ? is typed, an appropriate response is given, and the ? is not included in the atom buffer. (A ? can be included in the input text if it is preceded by a CTRL/V.)

20	.CMTAD	Parse a date and/or time field according to the setting of bits CM%IDA and CM%ITM. The user must input the field as requested. Any date format allowed by the IDTIM call can be input. If a date is not input, it is assumed to be the current date. If a time is not input, it is assumed to be 00:00:01. When both the date and time fields are input, they must be separated by one or more spaces. If the fields are input separately, they must be terminated with a space or carriage return. Word .CMDAT contains bits in the left half and an address in the right half as data for the function. The bits are: B0(CM%IDA) Parse a date B1(CM%ITM) Parse a time B2(CM%NCI) Do not convert the date and/or time to internal format. (Refer to Section 2.8.2.) The address in the right half is the beginning of a 3-word block in the caller's address space. On a successful return, this block contains data returned from the IDTNC call executed by COMND if B2(CM%NCI) was on in the COMND call (i.e., if the input date and/or time field was not to be converted to internal format). If B2(CM%NCI) was off in the COMND call, on a successful return, AC2 contains the internal date and time format.
21	.CMQST	Parse a quoted string up to the terminating quote. The delimiters for the string must be double quotation marks and are not copied to the atom buffer. A double quotation mark is input as part of the string if two double quotation marks appear together. This function is useful if the legal field terminators and the action characters are to be included as part of a string. The characters ?, ESC, and CTRL/F are not treated as action characters and are included in the string stored in the atom buffer. Carriage return is an invalid character in a quoted string and causes B1(CM%NOP) to be set on return.
22	.CMUQS	Parse an unquoted string up to one of the specified break characters. Word .CMDAT contains the address of a 4-word block of 128 break character mask bits. (Refer to word .RDBRK of the TEXTI call description for an explanation of the mask.) The characters scanned are not placed in the atom buffer. On return, .CMPTR is pointing to the break character. This function is useful for parsing a string with an arbitrary delimiter. The characters ?, ESC, and CTRL/F are not treated as action characters (unless they are specified in the mask) and can be included in the string. Carriage return can also be included if it is not one of the specified break characters.

23	.CMTOK	Parse the input and compare it with a given string. Word .CMDAT contains the pointer to the given string. This function sets B1(CM%NOP) in word .CMFLG of the command state block and returns if the next input characters do not match the given string. Leading blanks in the input are ignored. This function is useful for parsing single or multiple character operators (e.g., + or **).
24	.CMNUX	Parse a number and terminate on the first non-numeric character. Word .CMDAT contains the radix (from 2 to 10) of the number. On a successful return, AC2 contains the number. This function is useful for parsing a number that may not be terminated with a nonalphabetic character (e.g., 100PRINT FILEA). Note that non-numeric identifiers can begin with a digit (e.g., 1SMITH as a user name). When a non-numeric identifier and a number appear as alternates for a field, the order of the function descriptor blocks is important. The .CMNUX function, if given first, would accept the digit in the non-numeric identifier as a valid number instead of as the beginning character of a non-numeric identifier.
25	.CMACT	Parse an account string. The input, as delimited by the first nonalphanumeric character, is copied into the atom buffer; the delimiter is not copied. No verification is performed nor is any standard help message available.
26	.CMNOD	Parse a network node name. A node name consists of up to six alphanumeric characters. Lowercase characters are converted to uppercase characters. The input, as delimited by the first nonalphanumeric character, is copied into the atom buffer. Note that this function does not verify the existence of the node.

In addition to the .CMFNP word of the function descriptor block containing the function code in bits 0-8 (CM%FNC), this word also contains function-specific flag bits in bits 9-17 (CM%FFL) and the address of another function descriptor block in bits 18-35 (CM%LST).

The flag bits that can be set in bits 9-17 (CM%FFL) are as follows:

Bit	Symbol	Meaning
14	CM%PO	The field is to be parsed only and the field's existence is not to be verified. This bit currently applies to the .CMDIR and .CMUSR functions and is ignored for the remaining functions. On return, COMND sets B1(CM%NOP-no parse) only if the field typed is not in the correct syntax. Also, data returned in AC2 may not be correct.
15	CM%HPP	A pointer to a program-supplied help message for this field is given in word 2 (.CMHLP) of this function descriptor block.

16	CM%DPP	A pointer to a program-supplied default string for this field is given in word 3 (.CMDEF) of this function descriptor block.
17	CM%SDH	The output of the default help message is to be suppressed if the user types a question mark. (See below for the default messages.)

The address of another function descriptor block can be given in bits 18-35 (CM%LST) of the .CMFNP word. The use of this second descriptor block is described below.

Usually one COMND call is executed for each field in the command. However, for some fields, more than one type of input may be possible (e.g., after a keyword field, the next field could be a switch or a filename field). In these cases, all the possibilities for a field must be tried in an order selected to test unambiguous cases first.

When the COMND call cannot parse the field as indicated by the function code, it does one of two things:

1. It sets the current pointer and counts such that the next call will attempt to parse the same input over again. It then returns with B1(CM%NOP) set in the left half of the .CMFLG word in the command state block. The caller can then issue another COMND call with a function code indicating another of the possible fields. After the execution of each call, the caller should test the CM%NOP flag to see if the field was parsed successfully.

2. If an address of another function descriptor block is given in CM%LST, the COMND call moves to this descriptor block automatically and attempts to parse the field as indicated by the function code contained in B0-B8(CM%FNC) in word .CMFNP of that block. If the COMND call fails to parse the field using this new function code, it moves to a third descriptor block if one is given. This sequence continues until either the field is successfully parsed or the end of the chain of function blocks is reached. Upon completion of the COMND call, AC3 contains the addresses of the first and last function blocks used.

By specifying a chained list of function blocks, the program can have the COMND call automatically check all possible alternatives for a field and not have to issue a separate call for each one. In addition, if the user types a question mark, a list is output of all the alternatives for the field as indicated by the list of function descriptor blocks.

Word .CMHLP of the function descriptor block

This word contains a pointer to a program-supplied help text to be output if the user types a question mark when entering his command. The default help message is appended to the output of the program-supplied message if B17(CM%SDH) is not set. If B17(CM%SDH) is set, only the program-supplied message is output. If this word in the descriptor block is zero, only the default message is output when the user types a question mark. Bit 15(CM%HPP) must be set in word 0 (.CMFNP) of the function descriptor block for this pointer to be used.

The default help message depends on the particular function being used to parse the current field. The table below lists the default help message for each function available in the COMND call.

Default Help Messages

Function	Message
.CMKEY (keyword)	ONE OF THE FOLLOWING followed by the alphabetical list of valid keywords. If the user types a question mark in the middle of the field, only the keywords that can possibly match the field as currently typed are output. If no keyword can possibly match the currently typed field, the message KEYWORD (NO DEFINED KEYWORDS MATCH THIS INPUT) is output.
.CMNUM (number)	The help message output depends on the radix specified in .CMDAT in the descriptor block. If the radix is octal, the help message is OCTAL NUMBER If the radix is decimal, the help message is DECIMAL NUMBER If the radix is any other radix, the help message is A NUMBER IN BASE nn where nn is the radix.
.CMNOI (guide word)	None
.CMSWI (switch)	ONE OF THE FOLLOWING followed by the alphabetical list of valid switch keywords. The same rules apply as for .CMKEY function. (See above.)
.CMIFI (input file) .CMOFI (output file) .CMFIL (any file)	The help message output depends on the settings of certain bits in the GTJFN call. If bit GJ%OLD is off and bit GJ%FOU is on, the help message is OUTPUT FILESPEC Otherwise, the help message is INPUT FILESPEC
.CMFLD (any field)	None
.CMCFM (confirm)	CONFIRM WITH CARRIAGE RETURN
.CMDIR (directory)	DIRECTORY NAME
.CMUSR (user)	USER NAME
.CMCMA (comma)	COMMA
.CMINI (initialize)	None
.CMFLT (floating point)	NUMBER
.CMDEV (device)	DEVICE NAME
.CMTXT (text)	TEXT STRING

.CMTAD (date)	The help message depends on the bits set in .CMDAT in the descriptor block. If CM%IDA is set, the help message is DATE If CM%ITM is set, the help message is TIME If both are set, the help message is DATE AND TIME
.CMQST (quoted)	QUOTED STRING
.CMUQS (unquoted)	None
.CMTOK (token)	None
.CMNUX (number)	Same as .CMNUM
.CMACT (account)	None
.CMNOD (node)	NODE NAME

Word .CMDEF of the function descriptor block

This word contains a pointer to the ASCIZ string to be used as the default for this field. For this pointer to be used, bit 16 (CM%DPP) must be set in word 0 (.CMFNP) of the descriptor block. The string is output to the destination, as well as copied to the text buffer, if the user types an ESC or CTRL/F as the first non-blank character in the field. If the user types a carriage return, the string is copied to the atom buffer but is not output to the destination.

When the caller supplies a list of function descriptor blocks, the pointer for the default string must be included in the first block. The CM%DPP bit and the pointer for the default string are ignored when they appear in subsequent blocks. However, the default string can be worded so that it will apply to any of the alternative fields. The effect is the same as if the user had typed the given string.

Defaults for fields of a file specification can also be supplied with the .CMFIL function. If both the pointer to the default string and the GTJFN defaults have been provided, the COMND default will be used first and then, if necessary, the GTJFN defaults.

NOTE

> The function descriptor block, whose address is given in AC2, can be set up by the FLDDB. macro defined in MONSYM. (Refer to Appendix A for the definition of the FLDDB. macro.)

On a successful return, the COMND call returns flag bits in the left half of AC1 and preserves the address of the command state block in the right half of AC1. These flag bits are copied from word .CMFLG in the command state block and are described as follows.

Bits Returned on COMND Call

Bit	Symbol	Meaning
0	CM%ESC	An ESC was typed by the user as the terminator for this field.
1	CM%NOP	The field could not be parsed because it did not conform to the specified function(s). An error code is returned in AC2.
2	CM%EOC	The field was terminated with a carriage return.
3	CM%RPT	Characters already parsed need to be reparsed because the user edited them. This bit does not need to be examined if the program has supplied a reparse dispatch address in the right half of .CMFLG in the command state block.
4	CM%SWT	A switch field was terminated with a colon. This bit is on if the user either used recognition on a switch that ends with a colon or typed a colon at the end of the switch.
5	CM%PFE	The previous field was terminated with an ESC.

When a field cannot be parsed, B1(CM%NOP) is set in AC1, and one of the following error codes is returned in AC2. Note that if a list of function descriptor blocks is given and an error code is returned, the error is associated with the last function descriptor block in the list.

NPXAMB: ambiguous

NPXNSW: not a switch - does not begin with slash

NPXNOM: does not match switch or keyword

NPXNUL: null switch or keyword given

NPXINW: invalid guide word

NPXNC: not confirmed

NPXICN: invalid character in number

NPXIDT: invalid device terminator

NPXNQS: not a quoted string - does not begin with double quote

NPXNMT: does not match token

NPXNMD: does not match directory or user name

NPXCMA: comma not given

COMX18: invalid character in node name

COMX19: too many characters in node name

The COMND call causes other monitor calls to be executed, depending on the particular function that is requested. Failure of these calls usually results in the failure to parse the requested field. In these cases, the relevant error code can be obtained via the GETER and ERSTR monitor calls.

Any TBLUK error can occur on the keyword and switch functions.

Any NIN/NOUT and FLIN/FLOUT error can occur on the number functions.

Any GTJFN error except for GJFX37 can occur on the file specification functions.

Any IDTNC error can occur on the date/time function.

Any RCDIR or RCUSR error can occur on the directory and user functions.

Any STDEV error can occur on the device function.

Generates an illegal instruction interrupt on error conditions below.

COMND ERROR MNEMONICS:

COMNX1: invalid COMND function code

COMNX2: field too long for internal buffer

COMNX3: command too long for internal buffer

COMNX5: invalid string pointer argument

COMNX8: number base out of range 2-10

COMNX9: end of input file reached

COMX10: invalid default string

COMX11: invalid CMRTY pointer

COMX12: invalid CMBFP pointer

COMX13: invalid CMPTR pointer

COMX14: invalid CMABP pointer

COMX15: invalid default string pointer

COMX16: invalid help message pointer

COMX17: invalid byte pointer in function block

CRDIR JSYS 240

Creates, changes, or deletes a directory entry.

ACCEPTS IN AC1: pointer to ASCIZ string containing the structure and directory name.

AC2:

B0(CD%LEN)	set length of the argument block to the value given in word .CDLEN. This bit is not used in TOPS-20 version 3 and is provided for future releases.
B1(CD%PSW)	set password from argument block
B2(CD%LIQ)	set working disk storage limit from argument block
B3(CD%PRV)	set capability bits from argument block
B4(CD%MOD)	set mode bits from argument block
B5(CD%LOQ)	set permanent disk storage limit from argument block
B6(CD%NUM)	set directory number from argument block (valid only when creating a directory)
B7(CD%FPT)	set default file protection from argument block
B8(CD%DPT)	set directory protection from argument block
B9(CD%RET)	set default retention count from argument block
B10(CD%LLD)	set last LOGIN date from argument block
B11(CD%UGP)	set user groups from argument block
B12(CD%DGP)	set directory groups from argument block
B13(CD%SDQ)	set subdirectory quota from argument block
B14(CD%CUG)	set user groups assignable by this directory from argument block
B15(CD%DAC)	set default account from argument block
B17(CD%DEL)	delete this directory entry
B18-B35 (CD%APB)	address of the argument block

AC3: pointer to ASCIZ string containing the password of the directory. This pointer is required when a nonprivileged user is changing parameters for his directory.

RETURNS +1: always, with directory number in AC1

This monitor call requires the process to have WHEEL or OPERATOR capability enabled unless one of the following conditions is true:

1. The specified directory is one to which the caller has owner access and the caller is changing any one of the following parameters:

   password (.CDPSW)
   default file protection (.CDFPT)
   directory protection (.CDDPT)
   default retention count (.CDRET)
   default account (.CDDAC)

   This feature is installation dependent and is enabled by issuing function .SFCRD of the SMON monitor call.

2. The specified directory is inferior to the one to which the caller is currently connected and the caller has owner access to this inferior directory.

Refer to Section 2.2.6 for the description of owner access.

The format of the argument block is as follows:

Word	Symbol	Meaning	
0	.CDLEN	flag bits in the left half, and length of the argument block in the right half. The following bits are defined:	
		B0(CD%NSQ)	When restoring this directory, do not update its superior directory's quotas (permanent, working, and subdirectory quotas) to account for this directory. If this bit is off, the superior directory's quotas are updated. This bit is set by the DLUSER or DUMPER program to retain the superior directory's quotas when restoring its subdirectories. The process must have WHEEL or OPERATOR capability enabled to set this bit.
		B1(CD%NCE)	When restoring or reconstructing this directory, do not change any directory parameters if the directory currently exists on disk; set the parameters only if the directory does not exist. If this bit is off, the directory parameters as saved are restored for the directory. This bit is set by the DLUSER or DUMPER program to restore or reconstruct directories from out-of-date files without causing existing directories to revert to older parameters. The process must have WHEEL or OPERATOR capability enabled to set this bit.

1	.CDPSW	pointer to password string, which is a string from 1 to 39 alphanumeric characters.	
2	.CDLIQ	maximum number of pages that can be used for working disk storage (also known as logged-in quota).	
3	.CDPRV	capabilities for this user. (Refer to Section 2.6.1 for the capability bits.)	
4	.CDMOD	mode word.	
		B0(CD%DIR)	directory name can only be used for connecting to (i.e., the directory is a files-only directory). If this bit is off, the directory name can be used for logging in and connecting to.
		B1(CD%ANA)	accounts are alphanumeric. This bit is not used and is provided for compatibility with systems earlier than TOPS-20 version 3.
		B2(CD%RLM)	all the messages from the file <SYSTEM>MAIL.TXT are repeated each time the user logs in. If this bit is off, only the messages not previously printed are output when the user logs in.
5	.CDLOQ	maximum number of pages that can be used for permanent disk storage (also known as logged-out quota).	
6	.CDNUM	directory number, valid only when creating a directory. An error code is returned if user changes the number of an existing directory (CRDIX2) or gives a non-unique number (CRDIX8).	
7	.CDFPT	default file protection (18 bits, right-justified).	
10	.CDDPT	directory protection (18 bits, right-justified).	
11	.CDRET	default number of generations of a file to be retained in the directory (retention count). Valid numbers are 0 to 63, with 0 being an infinite number.	
12	.CDLLD	date of last login.	
13	.CDUGP	pointer to user group list.	
14	.CDDGP	pointer to directory group list.	
15	.CDSDQ	maximum number of directories that can be created inferior to this directory. This parameter allows a user to create directories with the BUILD command.	

16	.CDCUG	pointer to user group list. This list contains the group numbers that can be given to directories inferior to this one.
17	.CDDAC	pointer to default account for this user.

The format of each group list is a table with the first word containing a count of the number of words (including the count word) in the table and each subsequent word containing a group number.

When CRDIR is being executed to create a directory, bits 0-17 of AC2 may optionally be on or off. In particular if the following bits of AC2 are off, CRDIR will default fields to:

B2(CD%LIQ)	maximum working disk file storage to 250 pages
B3(CD%PRV)	no special capabilities
B4(CD%MOD)	directory name that can be used for logging in and that lists the messages from <SYSTEM>MAIL.TXT only once
B5(CD%LOQ)	maximum permanent disk file storage to 250 pages
B6(CD%NUM)	the first unused directory number. B6 should normally be off.
B7(CD%FPT)	default file protection to 777700
B8(CD%DPT)	directory protection to 777700
B9(CD%RET)	default file retention count to 1
B10(CD%LLD)	never logged in
B11(CD%UGP)	no user groups
B12(CD%DGP)	no directory groups
B13(CD%SDQ)	no ability to create inferior directories
B14(CD%CUG)	no assignable user groups for inferior directories
B15(CD%DAC)	no default account

When CRDIR is being executed to change a directory and any of B0-B17 of AC2 is off, the corresponding parameter is not affected.

When CRDIR is being executed to delete a directory, the settings of B0-B17 of AC2 are ignored. A CRDIR call cannot be given to delete a directory that has directories inferior to it.

The GTDIR call can be used to obtain the directory information.

Generates an illegal instruction interrupt on error conditions below.

CRDIR ERROR MNEMONICS:

CRDIX1: WHEEL or OPERATOR capability required

CRDIX2: illegal to change number of old directory

CRDIX3: insufficient system resources (Job Storage Block full)

CRDIX4: superior directory full

CRDIX5: directory name not given

CRDIX6: directory file is mapped

CRDIX7: file(s) open in directory

CRDIX8: invalid directory number

CRDIX9: internal format of directory is incorrect

CRDI10:	maximum directory number exceeded; index table needs expanding
CRDI11:	invalid terminating bracket on directory
CRDI12:	structure is not mounted
CRDI13:	request exceeds superior directory working quota
CRDI14:	request exceeds superior directory permanent quota
CRDI15:	request exceeds superior directory subdirectory quota
CRDI16:	invalid user group
CRDI17:	illegal to create non-files-only subdirectory under files only directory
CRDI18:	illegal to delete logged-in directory
CRDI19:	illegal to delete connected directory
CRDI20:	WHEEL, OPERATOR, or requested capability required
CRDI21:	working space insufficient for current allocation
CRDI22:	subdirectory quota insufficient for existing subdirectories
CRDI23:	superior directory does not exist
CRDI24:	invalid subdirectory quota

CRJOB JSYS 2

Creates a new job and optionally logs it in. This monitor call causes the functions that are normally performed when a job is created (e.g., assignment of a JSB, the primary I/O designators, and the job controlling terminal) to be performed for the new job.

Refer to the TOPS-20AN Monitor Calls User's Guide for more information on the CRJOB monitor call.

ACCEPTS IN	AC1:	flag bits
	AC2:	address of argument block
	AC3:	number of job whose ownership is being released
RETURNS	+1:	failure, with error code in AC1
	+2:	success, with the number of the new job in AC1

The flag bits defined in the left half of AC1 are as follows:

Bit	Symbol	Meaning
0	CJ%LOG	Log in the new job. If this bit is off, the new job is created but not logged in.
1	CJ%NAM	Set the user name and password from the argument block. If this bit is off, the user name of the caller is given to the new job.
2-3	CJ%ACT	Set the account of the new job to the following:

Code	Symbol	Meaning
0	.CJUCA	Use current account of caller.
1	.CJUAA	Use account from the argument block.
2	.CJUDA	Use default account of user whose job is being created.

Bit	Symbol	Meaning
4	CJ%ETF	Place the Command Language Processor in the top-level process of the new job. If this bit is off and B5(CJ%FIL) is on, the file pointed to in the argument block is placed in the top-level process of the new job.
5	CJ%FIL	Move (via a GET call) the file pointed to in the argument block into a process in the new job. If B4(CJ%ETF) is off, the file is placed in the top-level process of the new job. If B4(CJ%ETF) is on, the file is placed in the process designated in the Command Language Processor's PRARG argument block. (See below.) If B5(CJ%FIL) is off, no file is moved into a process of the new job, and the top-level process of the new job is the Command Language Processor.
6	CJ%ACS	Load the ACs from the value in the argument block. The ACs are loaded only if a program other than the Command Language Processor is being run.
7	CJ%OWN	Maintain ownership of the new job. This means that the new job cannot be logged out until the caller releases ownership of it. If this bit is off, the caller does not have ownership of the new job.
8	CJ%WTA	Do not start the new job until it is attached to a terminal. If this bit is off, the new job is started.
9	CJ%NPW	Do not check the password given when the new job is logged in. If this bit is off, the password is checked unless the new job is being logged in with the same user name as the caller or with WHEEL or OPERATOR capability enabled.

10	CJ%NUD	Do not update the date of LOGIN for user logging in to the new job. If this bit is off, the date of LOGIN is updated unless the user is logging in with the same user name as the caller or with WHEEL or OPERATOR capability enabled.
11	CJ%SPJ	Set (via a SPJFN call) the primary input and output designators from the argument block before starting the job. The primary I/O designators are not changed for a Command Language Processor in the top-level process of the new job; they are changed only for inferior processes. If this bit is off, the primary I/O designators of the new job are the job's controlling terminal.
12	CJ%CAP	Set the new job's allowed user capabilities (right half) to be the same as the caller's currently enabled capabilities, until the new job is logged in. If this bit is off, the new job has the user capabilities associated with the user whose job is being created.
13	CJ%CAM	Set the new job's allowed capabilities to be the combination of (AND function) the capability mask in the argument block and the new job's user capabilities. If this bit is off, the new job has the capabilities associated with the user whose job is being created.
14	CJ%SLO	Send a IPCF message to the PID supplied in argument block when the new job is logged out. If this bit is off, no message is sent when the new job is logged out.
17	CJ%DSN	Release ownership of the previously created job whose number is in AC3. If this bit is on, it overrides the setting of all other bits in AC1, and no change is made to the job's status other than the change in ownership.

The format of the argument block (whose address is given in AC2) is as follows:

Word	Symbol	Meaning
0	.CJNAM	Pointer to the user name string.
1	.CJPSW	Pointer to the password string.
2	.CJACT	Pointer to the account string or 5B2+numeric account.
3	.CJFIL	Pointer to the name of the file to be moved (via a GET call) into a process of the new job. The new job must have read access to the file. The process into which the file is placed depends on the setting of B4(CJ%ETF).
4	.CJSFV	Offset in the entry vector to use as the start address of the file pointed to by word .CJFIL. This offset is the argument to the SFRKV call used to start the process.

5	.CJTTY	Terminal designator of the new job's controlling terminal. This terminal must be assigned by the caller. The terminal is then released and assigned to the new job. If the new job is to be detached, the .NULIO designator (377777) is given.
6		Reserved for the future and must be zero.
7	.CJACS	Address of a 16-word block whose contents are to be loaded in the new job's ACs if a program other than the Command Language Processor is being run.
10	.CJEXF	Flag bits to be passed to the Command Language Processor in the top-level process of the new job. The bits are: B0 Suppress the herald printed by the Command Language Processor. B1 Move the file pointed to by word .GJFIL into the process whose handle is in the PRARG block. (See below.) B2 Start the process at the offset in the entry vector given in word .CJSFV. This process is started after the Command Language Processor is initialized. B3 Output the text printed when a LOGIN command is given (e.g., system messages, job number, terminal number). This word is copied into the PRARG argument block passed to the Command Language Processor. (See Below.)
11	.CJPRI	Primary input and output designators for the inferior processes of the new job. These designators must refer to device designators. The Command Language Processor in the top-level process of the new job executes a SPJFN call to set these designators.
12	.CFCPU	Runtime limit for the new job. When this limit is reached, an interrupt is generated (via a TIMER call), and the Command Language Processor will execute a LGOUT call for the new job. A zero in this word means there is no runtime limit on the job.
13	.CJCAM	Capability mask for the new job.
14	.CJSLO	PID to which an IPCF message is to be sent when the new job is logged out.

When the Command Language Processor is the top-level process of the new job, it is started at its normal start address (i.e., at the first word of its entry vector). An argument block is created by the CRJOB call and passed to this process via the PRARG monitor call. The format of this PRARG argument block is

Word	Meaning
0	Count of words in the block, not including this word.
1	Bit pattern identifying the block as a CRJOB-created block.
2	Pointer to the first argument being passed. This argument is a copy of the flag bits from word .CJEXF of the argument block given on the CRJOB call.
3	Pointer to the second argument being passed. This argument contains information about the process being started (a process handle in the left half of the word and the offset in its entry vector in the right half of the word).

The CRJOB call causes other monitor calls to be executed, depending on the particular function that is performed.

Any GTJFN and OPENF errors can occur when obtaining the specified file.

Any SFRKV error can occur when starting the program in the specified file.

Any LOGIN and account validation errors can occur when logging in the job.

CRJOB ERROR MNEMONICS:

CRJBX1: invalid parameter or function bit combination

CRJBX2: illegal for created job to enter MINI-EXEC

CRJBX4: terminal is not available

CRJBX5: unknown name for LOGIN

CRJBX6: insufficient system resources

CRLNM JSYS 502

Defines or deletes a logical name assignment. Logical names are used to specify a set of default values for each field requested by a GTJFN monitor call. When a logical name is passed to the GTJFN call, any fields not specified by the user are supplied from the fields defined in the logical name definition. (Refer to Section 2.2.2 and to the INLNM and LNMST monitor call descriptions for more information on logical names.)

ACCEPTS IN	AC1:	function code
	AC2:	pointer to the logical name
	AC3:	pointer to the logical name definition string
RETURNS	+1:	failure, error code in AC1
	+2:	success, updated string pointer in AC3

The codes for the functions are as follows:

0	.CLNJ1	delete one logical name from the job
1	.CLNS1	delete one logical name from the system
2	.CLNJA	delete all logical names from the job
3	.CLNSA	delete all logical names from the system
4	.CLNJB	create a logical name for the job
5	.CLNSY	create a logical name for the system

CRLNM ERROR MNEMONICS:

CRLNX1:	logical name is not defined
CRLNX2:	WHEEL or OPERATOR capability required
CRLNX3:	invalid function
GJFX4:	invalid character in filename
GJFX5:	field cannot be longer than 39 characters
GJFX6:	device field not in a valid position
GJFX7:	directory field not in a valid position
GJFX8:	directory terminating delimiter is not preceded by a valid beginning delimiter
GJFX9:	more than one name field is not allowed
GJFX10:	generation number is not numeric
GJFX11:	more than one generation number field is not allowed
GJFX12:	more than one account field is not allowed
GJFX13:	more than one protection field is not allowed

GJFX14: invalid protection

GJFX15: invalid confirmation character

GJFX22: insufficient system resources (Job Storage Block full)

GJFX31: invalid wildcard designator

DEBRK JSYS 136

Dismisses the current software interrupt routine in progress and resumes the process at the location specified by the PC stored in the priority level table. (Refer to Section 2.5.7.)

RETURNS +1: if no software interrupt is currently in progress and if an ERJMP or ERCAL instruction follows the DEBRK

Generates an illegal instruction interrupt on error conditions below.

DEBRK ERROR MNEMONICS:

DBRKX1: no interrupts in progress

DELDF JSYS 67

Reclaims space by expunging disk files that have been marked for deletion with DELF. This call first checks the user's access to the directory, verifying that the user is allowed to expunge files from it.

ACCEPTS IN AC1: B0(DD%DTF) delete temporary files (;T) also

B1(DD%DNF) delete nonexistent files that are not now opened

B2(DD%RST) rebuild the symbol table

B3(DD%CHK) check internal consistency of directory. If an error occurs, the symbol table should be rebuilt. If B2(DD%RST) is also set, it is ignored, and the DELDF call must be executed again with B2(DD%RST) on to rebuild the symbol table.

AC2: directory number

RETURNS +1: always

The directory number given must be that of the user's connected or logged-in directory unless the process has WHEEL or OPERATOR capability enabled.

If errors still occur after the symbol table is rebuilt, the process should restore the directory from magnetic tape or the user should request help from the operator.

Generates an illegal instruction interrupt on error conditions below.

DELDF ERROR MNEMONICS:

DELDX1:	WHEEL or OPERATOR capability required
DELDX2:	invalid directory number
DELFX2:	file cannot be expunged because it is currently open
DELFX4:	directory symbol table could not be rebuilt
DELFX5:	directory symbol table needs rebuilding
DELFX6:	internal format of directory is incorrect
DELFX7:	FDB formatted incorrectly; file not deleted
DELFX8:	FDB not found; file not deleted

DELF JSYS 26

Deletes the specified disk file and (if the file is closed) releases the JFN. The file is not expunged immediately but is marked for later expunging by the system or with the DELDF or LGOUT monitor call.

ACCEPTS IN AC1:

B0(DF%NRJ)	do not release the JFN
B1(DF%EXP)	expunge the contents of the file. This also deletes the FDB entry in the directory. B0(DF%NRJ) and B1(DF%EXP) cannot be set simultaneously.
B2(DF%FGT)	expunge the file but do not deassign its addresses. The process must have WHEEL or OPERATOR capability enabled to set this bit. This bit should be set only by an operator or system specialist to delete a file that has a damaged or inconsistent index block.
B3(DF%DIR)	delete and expunge a directory file. The process must have WHEEL or OPERATOR capability enabled to set this bit. This bit should be set only by an operator or specialist to delete a bad directory.
B18-B35 (DF%JFN)	JFN of the file being deleted

RETURNS

+1:	failure, error code in AC1
+2:	success, JFN is released unless B0(DF%NRJ) is on or the file is opened.

By setting B0(DF%NRJ), the user can delete multiple files by giving a JFN to GNJFN that represents a group of files and processing each file in the group.

The DELF call takes the +1 return if the JFN is assigned to a non-directory device.

DELF ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX7:	JFN cannot refer to output wildcard designators
DESX9:	invalid operation for this device
DELFX1:	delete access required
DELFX2:	file cannot be expunged because it is currently opened
DELFX3:	system scratch area depleted; file not deleted
DELFX4:	directory symbol table could not be rebuilt
DELFX5:	directory symbol table needs rebuilding
DELFX6:	internal format of directory is incorrect
DELFX7:	FDB formatted incorrectly; file not deleted
DELFX8:	FDB not found; file not deleted
DELFX9:	file is not a directory file
DELF10:	directory still contains subdirectory
DLFX10:	cannot delete directory; file still mapped
DLFX11:	cannot delete directory file in this manner
WHELX1:	WHEEL or OPERATOR capability required

DELNF JSYS 317

Deletes all but the specified number of generations of a disk file. The files are marked for deletion and are expunged at a later time either automatically by the system or explicitly with the DELDF or LGOUT call.

ACCEPTS IN AC1: JFN of the file

AC2: the number of generations to retain

RETURNS +1: failure, error code in AC1

+2: success, with the number of files deleted in AC2

Starting at the file specified by the JFN, the DELNF call decrements the generation number, first retaining the specified number of generations before deleting the remaining generations.

DELNF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

DELFX1: delete access required

DEQ JSYS 514

Removes a request for a specific resource from the queue associated with that resource. The request is removed whether or not the process has a lock for the resource or is only waiting in the queue for the resource.

This call can be used to remove any number of requests. If one of the requests cannot be dequeued, the dequeueing procedure continues until all requests that can be have been dequeued. An error return is given for the last request found that could not be dequeued. The process can then execute the ENQC call to determine the current status of each request. However, if the process attempts to dequeue more pooled resources than it originally allocated, the error return is taken and none of the pooled resources are dequeued.

Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

ACCEPTS IN AC1: function code

AC2: address of argument block (required only for the .DEQDR function)

RETURNS +1: failure, error code in AC1

+2: success

The available functions are as follows:

Code	Symbol	Meaning
0	.DEQDR	Remove the specified requests from the queue. This function is the only one requiring an argument block.
1	.DEQDA	Remove all requests for this process from the queues. This action is taken on a RESET or LGOUT call. The error return is taken if the process has not given an ENQ call.
2	.DEQID	Remove all requests that correspond to the specified request identifier(ID). This function allows the process to release a class of locks in one call without itemizing each lock in an argument block. It is useful when dequeueing in one call the same locks that were enqueued in one call. To use this function, the process places the 18-bit request ID in AC2.

The format of the argument block for function .DEQDR is identical to that given on the ENQ call. (Refer to the ENQ monitor call description.) However, the .ENQID word of the argument block is not used on a DEQ call and must be zero.

DEQ ERROR MNEMONICS:

ENQX1: invalid function

ENQX2: level number too small

ENQX3: request and lock level numbers do not match

ENQX4: number of pool and lock resources do not match

ENQX6: requested locks are not all locked

ENQX7: no ENQ on this lock

ENQX9: invalid number of blocks specified

ENQX10: invalid argument block length

ENQX11: invalid software interrupt channel number

ENQX13: indirect or indexed byte pointer not allowed

ENQX14: invalid byte size

ENQX15: ENQ/DEQ capability required

ENQX16: WHEEL or OPERATOR capability required

ENQX17: invalid JFN

ENQX18: quota exceeded

ENQX19: string too long

ENQX20: locked JFN cannot be closed

ENQX21: job is not logged in

DESX8: file is not on disk

DEVST JSYS 121

Translates the given device designator to its corresponding string.

ACCEPTS IN AC1: destination designator

AC2: device designator

RETURNS +1: failure, error code in AC1

+2: success, updated string pointer in AC1, if pertinent

The STDEV monitor call can be used to translate a string to its corresponding device designator.

DEVST ERROR MNEMONICS:

DEVX1: invalid device designator

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

IOX11: quota exceeded or disk full

DFIN JSYS 234

Inputs a double-precision, floating-point number. Currently this call inputs a KA10 format double-precision floating-point number.

ACCEPTS IN AC1: source designator

RETURNS +1: failure, error code in AC4 and updated string pointer in AC1, if pertinent.

+2: success, double-precision, floating-point number (extended format where necessary) in AC2 and AC3 and updated string pointer in AC1, if pertinent.

DFIN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

FLINX1: first character is not blank or numeric

FLINX2: number too small

FLINX3: number too large

FLINX4: invalid format

DFOUT JSYS 235

Outputs a double-precision, floating-point number. Currently this call outputs a KA10 format double-precision, floating-point number.

ACCEPTS IN AC1: destination designator

AC2: a normalized double-precision

AC3: floating-point number in either extended or non-extended range.

AC4: format control word. (Refer to Section 2.8.1.2.)

RETURNS +1: failure, error code in AC4 and updated string pointer in AC1, if pertinent.

+2: success, updated string pointer in AC1, if pertinent.

DFOUT ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

FLOTX1: column overflow in field 1 or 2

FLOTX2: column overflow in field 3

FLOTX3: invalid format specified

IOX11: quota exceeded or disk full

DIAG JSYS 530

Reserves a channel and either a single device or all devices attached to that channel. This call is also used to release the channel and its devices. When the request is made, no new activity is initiated on the requested channel, and the monitor waits for current activity on all devices connected to the channel to complete. When the channel becomes idle, the process requesting the channel continues running.

ACCEPTS IN AC1: length of the argument block in the left half, and address of the argument block in the right half.

RETURNS +1: failure, error code in AC1

+2: success

The format of the argument block is as follows:

function code
data words for the function

The available functions are as follows:

Function	Symbol	Data Words	Meaning
1	.DGACU	device address time limit in milliseconds	Assign the channel and a single device. Force the device to be released after the time limit specified.
2	.DGACH	device address	Assign the channel and all devices.
3	.DGRCH	device address	Release the channel and all assigned devices.
4	.DGSCP	device address channel control word	Set up the channel program. The data transfer must be in one page. The user page pointed to by the channel control word is locked in memory. The Exec Process Table location corresponding to the channel is updated with the appropriate physical address channel control word.
5	.DGRCP	device address	Release the channel program. The page pointed to by the channel control word for the specified channel is unlocked. This function is not required before specifying a new channel program.
6	.DGGCS	device address word 0 word 1 word 2 word 4	Return the status of the channel. The specified words are the logout area for the channel.

The device address given in the argument block is a machine-dependent specification for the channel and device to be assigned. The devices that can be assigned must be attached to the RH20 controller and must be mounted by a process with the WHEEL, OPERATOR, or MAINTENANCE capability enabled. The format of the device address word is

```
 0          2 3        9 10      23 24      29 30          35
!=========================================================!
!  address  ! device  !    0     !  unit   !   subunit   !
!   type    !  code   !          !         !             !
!=========================================================!
```

DIAG ERROR MNEMONICS:

DIAGX1: invalid function

DIAGX2: device is not assigned

DIAGX3: argument block too small

DIAGX4: invalid device type

DIAGX5: WHEEL, OPERATOR, or MAINTENANCE capability required

DIAGX6: invalid channel command list

DIAGX7: illegal to do I/O across page boundary

DIAGX8: no such device

DIAGX9: unit does not exist

DIAG10: subunit does not exist

DIBE JSYS 212

Dismisses the process until the designated file input buffer is empty.

ACCEPTS IN AC1: file designator

RETURNS +1: always

Returns immediately if the designator is not associated with a terminal.

The DOBE monitor call can be used to dismiss the process until the designated file output buffer is empty.

Generates an illegal instruction interrupt on error conditions below.

DIBE ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

DIC JSYS 133

Deactivates the specified software interrupt channels. (Refer to Section 2.5.1.)

ACCEPTS IN AC1: process handle

AC2: 36-bit word
Bit n means deactivate channel n

RETURNS +1: always

Software interrupt requests to deactivated channels are ignored except for interrupts generated on panic channels. Panic channel interrupts are passed to the closest superior process that has the specific channel enabled.

The AIC monitor call is used to activate specified software interrupt channels.

Generates an illegal instruction interrupt on error conditions below.

DIC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

DIR JSYS 130

Disables the software interrupt system for a process.

ACCEPTS IN AC1: process handle

RETURNS +1: always

If software interrupt requests are generated while the interrupt system is disabled, the requests are remembered and take effect when the interrupt system is reenabled unless an intervening CIS call is executed. However, interrupts on panic channels will still be generated even though the system is disabled. In addition, if the CTRL/C terminal code is assigned to a channel, it will still generate an interrupt that cannot be disabled with a DIR call. CTRL/C interrupts can be disabled by deactivating the channel to which the code is assigned or by the monitor.

The EIR monitor call can be used to enable the software interrupt system for a process.

Generates an illegal instruction interrupt on error conditions below.

DIR ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

DIRST JSYS 41

Translates the specified 36-bit user or directory number to its corresponding string and writes it to the given destination. When a user number is given, the string returned is the corresponding user name without any punctuation. When a directory number is given, the string returned is the corresponding structure and directory name including punctuation (i.e., structure:<directory>).

ACCEPTS IN AC1: destination designator

AC2: user or directory number

RETURNS +1: failure, with error code in AC1.

+2: success, string written to destination, updated string pointer, if pertinent, in AC1

The RCDIR monitor call can be used to translate a directory string to its corresponding directory number. The RCUSR monitor call can be used to translate a user name string to its corresponding user number.

DIRST ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

DELFX6: internal format of directory is incorrect

DIRX1: invalid directory number

DIRX2: insufficient system resources

DIRX3: internal format of directory is incorrect

STRX01: structure is not mounted

IOX11: quota exceeded or disk full

DISMS JSYS 167

Dismisses this process for the specified amount of time.

ACCEPTS IN AC1: number of milliseconds for which the process is to be dismissed

RETURNS +1: when the elapsed time is up

The maximum number of milliseconds that a process can be dismissed is 2^26 milliseconds. If a number is given that is greater than the maximum, it is ignored and the maximum dismiss time is given. The time resolution is limited to the scheduling frequency (about 20 milliseconds).

DOBE JSYS 104

Dismisses the process until the designated file output buffer is empty.

ACCEPTS IN AC1: destination designator

RETURNS +1: always

Returns immediately if designator is not associated with a terminal.

The DIBE monitor call can be used to dismiss the process until the designated file input buffer is empty.

Generates an illegal instruction interrupt on error conditions below.

DOBE ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

DSKAS JSYS 244

Assigns or deassigns specific disk addresses. This monitor call requires the process to have WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1:

	B0(DA%DEA)	deassign the specified address. If the address is currently assigned, control returns to the next instruction following the call (+1 return). If the address was not previously assigned, a BUGCHK occurs.
	B1(DA%ASF)	assign a free page near the specified address. Assignment is on the same cylinder as the specified address, if possible, or on a nearby cylinder. If the specified address is 0, a page is assigned on a cylinder that is at least one-half free. If the assignment is not possible because the disk is full, control returns to the next instruction following the call.
	B2(DA%CNV)	convert the specified address according to the setting of B3(DA%HWA).
	B3(DA%HWA)	the specified address is a hardware address. If this bit if off, the specified address is a software address.
	B4(DA%INI)	initialize a private copy of the bit table.
	B5(DA%WRT)	write the private copy of the bit table to a new bit table file.
	B18-B35 (DA%ADR)	disk address
AC2:	device designator of structure. If DA%CNV is on in AC1, this argument is not required.	

RETURNS

+1:	failure, address already assigned or cannot be assigned
+2:	success, address assigned in AC1

Generates an illegal instruction interrupt on error conditions below.

DSKAS ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

DSKOP JSYS 242

Allows the process to reference physical disk addresses when performing disk transfers. This monitor call requires the process to have WHEEL, OPERATOR, or MAINTENANCE capability enabled to read and write data. However, a process with only MAINTENANCE capability enabled can write data only if it is using physical addresses (.DOPPU) and writing to a unit that is not part of a mounted structure.

```
ACCEPTS IN AC1:  B0-B1(DOP%AT)  field indicating the address type.
                                For physical channel and unit
                                addresses, the value of the field is
                                1(.DOPPU) and the remainder of AC1 is
                                   B2-B6(DOP%CN) channel number
                                   B7-B12(DOP%UN) unit number
                                   B13-B35(DOP%UA) unit address
                                For a structure and a relative
                                address, the value of the field is
                                2(.DOPSR) and the remainder of AC1 is
                                   B2-B10(DOP%SN) structure designator
                                     flag (0 is structure PS:). A
                                     value of -1 means the structure is
                                     indicated by the structure
                                     designator (refer to Section 2.4)
                                     in AC4.
                                   B11-B35(DOP%RA) relative address

                                Any other values for this field are
                                illegal.

           AC2:  control flags in the left half and a count of the
                 number of words to transfer in the right half. The
                 control flags are

                 B11(DOP%IL)   inhibit error logging
                 B12(DOP%IR)   inhibit error recovery
                 B14(DOP%WR)   write data to the disk. If this bit is
                               off, read data from the disk.
                 B18-B35       word count. Since the transfer cannot
                 (DOP%CT)      cross a page boundary, the count must be
                               less than or equal to 1000 (octal)
                               words.

           AC3:  address in caller's address space from which data is
                 written or into which data is read.

           AC4:  device designator of the structure. This word is
                 used if the value given for DOP%SN is -1.

RETURNS     +1:  always, AC1 is nonzero if an error occurred or zero
                 if no error occurred.
```

If an error occurs and DOP%IL is on in the call, no error logging is performed. If DOP%IL is off, the standard system error logging is performed.

If an error occurs and DOP%IR is on in the call, no retries or ECC corrections, if applicable, are attempted. If DOP%IR is off, the standard system error recovery procedure is followed.

Generates an illegal instruction interrupt on error conditions below.

DSKOP ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

DSKOX1: channel number too large

DSKOX2: unit number too large

DSKOX3: invalid structure number

DSKOX4: invalid address type specified

DTACH JSYS 115

Detaches the controlling terminal from the current job. (The ATACH call with bit 1 (AT%NAT) of AC2 set can be used to detach a job other than the current job.) A console-detached entry is appended to the accounting data file.

RETURNS +1: always

The DTACH call is a no-op if the job is already detached.

The ATACH monitor call is used to attach the controlling terminal to a specified job.

DTI JSYS 140

Deassigns a terminal code.

ACCEPTS IN AC1: terminal code; refer to Section 2.5.6

RETURNS +1: always

The DTI call is a no-op if the specified terminal code was not assigned by the current process.

The ATI monitor call is used to assign a terminal code.

Generates an illegal instuction interrupt on error conditions below.

DTI ERROR MNEMONICS:

TERMX1: invalid terminal code

DUMPI JSYS 65

Reads data words into memory in unbuffered data mode. The file must be open for data mode 17. (Refer to Section 2.4.2.2 for information about unbuffered magnetic tape I/O.)

ACCEPTS IN AC1: JFN

AC2: B0(DM%NWT) do not wait for completion of requested operation

B18-B35 (DM%PTR) address of command list in memory

RETURNS +1: failure, error code in AC1, pointer to offending command in AC2

+2: success, pointer in AC2 updated to last command

The use of B0(DM%NWT) allows data operations to be double buffered with a resulting increase in speed. When this bit is on, DUMPI/DUMPO returns immediately after the request is queued. This allows the program to overlap computations with I/O transfers. If the second request is then made, the program is blocked until the first request is completed. Generally, for a sequence of overlapped DUMPI/DUMPO calls, return from the Nth call indicates that the Nth-1 request has completed and that the Nth request is now in progress. This bit is implemented only for magnetic tape.

The GDSTS call can be used after the transfer is completed to determine the number of words read.

If an error occurs on the Nth request, the failure return is given on the Nth+1 call, and the Nth+1 request is ignored. This means that the program will discover an error on a request only after making the next request. The next request is ignored to prevent improper operation and must be reissued after the error has been processed. The GDSTS call can be executed to determine the cause for the error.

COMMAND LIST FORMAT

Three types of entries may occur in the command list.

1. IOWD n, loc - Causes n words from loc through loc+n-1 to be transferred. The next command is obtained from the location following the IOWD. Each IOWD word must be read or written with a separate record from magnetic tape. The IOWD pseudo-op generates XWD -n,loc-1.

2. XWD 0, y - Causes the next command to be taken from location y. Referred to as a GOTO word.

3. 0 - Terminates the command list.

DUMPI ERROR MNEMONICS:

DUMPX1: command list error

DUMPX2: JFN is not open in dump mode

DUMPX3: address error (too big or crosses end of memory)

DUMPX4:	access error (cannot read or write data in memory)
DUMPX5:	no-wait dump mode not supported for this device
DUMPX6:	dump mode not supported for this device
DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX5:	file is not open
IOX1:	file is not opened for reading
IOX4:	end of file reached
IOX5:	device or data error

DUMPO JSYS 66

Writes data words from memory in unbuffered data mode. The file must be open for data mode 17. (Refer to Section 2.4.2.2 for information about unbuffered magnetic tape I/O.)

ACCEPTS IN	AC1:	JFN	
	AC2:	B0(DM%NWT)	do not wait for completion of requested operation
		B18-B35 (DM%PTR)	address of command list in memory
RETURNS	+1:	failure, error code in AC1, pointer to offending command in AC2	
	+2:	success, pointer in AC2 updated to last command	

The use of B0(DM%NWT) allows data operations to be double buffered with a resulting increase in speed. When this bit is on, DUMPI/DUMPO returns immediately after the request is queued. This allows the program to overlap computations with I/O transfers. If the second request is then made, the program is blocked until the first request is completed. Generally, for a sequence of overlapped DUMPI/DUMPO calls, return from the Nth call indicates that the Nth-1 request has completed and that the Nth request is now in progress. This bit is implemented only for magnetic tape.

Refer to the DUMPI description for the command list format.

The GDSTS call can be used after the transfer is completed to determine the number of words written.

DUMPO ERROR MNEMONICS:

DUMPX1:	command list error
DUMPX2:	JFN is not open in dump mode
DUMPX3:	address error (too big or crosses end of memory)
DUMPX4:	access error (cannot read or write data in memory)
DUMPX5:	no-wait dump mode not supported for this device
DUMPX6:	dump mode not supported for this device
DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX5:	file is not open
IOX2:	file is not opened for writing
IOX5:	device or data error
IOX11:	quota exceeded or disk full

DVCHR JSYS 117

Returns the device characteristics of the specified device.

ACCEPTS IN AC1: JFN or device designator

RETURNS +1: always, with

AC1 containing the device designator (even if a JFN was given).
AC2 containing the device characteristics word.
AC3 containing the job number to which the device is assigned in the left half and the unit number in the right half. If the device is a structure or does not have units, the right half is -1.

The contents of AC3 are -1 if the device is not assigned to any job.

Device Characteristics Word

Bit	Symbol	Meaning
0	DV%OUT	device can do output
1	DV%IN	device can do input
2	DV%DIR	device has a directory
3	DV%AS	device is assignable with ASND
4	DV%MDD	device has multiple directories

Device Characteristics Word (Cont.)

5	DV%AV	device is available or assigned to this job		
6	DV%ASN	device is assigned by ASND		
8	DV%MNT	device is mounted		
9-17	DV%TYP	device type		
		0	.DVDSK	disk
		2	.DVMTA	magnetic tape
		7	.DVLPT	line printer
		10	.DVCDR	card reader
		11	.DVFE	front-end pseudo-device
		12	.DVTTY	terminal
		13	.DVPTY	pseudo-terminal
		15	.DVNUL	null device
		16	.DVNET	ARPA network
20-35	DV%MOD	data mode in which device can be opened		
		B20	DV%M17	dump mode
		B27	DV%M10	image mode
		B35	DV%M0	normal mode

Generates an illegal instruction interrupt on error conditions below.

DVCHR ERROR MNEMONICS:

DEVX1: invalid device designator

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

EFACT JSYS 5

Makes an entry in the FACT file. The EFACT monitor call is obsolete and provided only for existing programs that make entries in the FACT file. New programs should use the USAGE monitor call to make entries in the new USAGE file.

ACCEPTS IN AC1: LH: negative size of entry
RH: pointer to beginning of entry (size bits of entry will be updated by the system from the negative size specified)

RETURNS +1: failure, error code in AC1

+2: success

The EFACT call returns successfully without making an entry in the FACT file if the monitor flag SF%FAC (refer to SMON and TMON calls) is not set.

The EFACT monitor call can be executed only by the monitor or by a process that has WHEEL or OPERATOR capability enabled.

EFACT ERROR MNEMONICS:

EFCTX1: WHEEL or OPERATOR capability required

EFCTX2: entry cannot be longer than 64 words

EFCTX3: fatal error when accessing FACT file

EIR JSYS 126

Enables the software interrupt system for a process. (Refer to Section 2.5.)

ACCEPTS IN AC1: process handle

RETURNS +1: always

The DIR monitor call can be used to disable the software interrupt system for a process.

Generates an illegal instruction interrupt on error conditions below.

EIR ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

ENQ JSYS 513

Requests access to a specific resource by placing a request in the queue for that resource. This call can be used to request any number of resources.

Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

ACCEPTS IN AC1: function code

AC2: address of argument block

RETURNS +1: failure, error code in AC1

+2: success

The available functions are as follows:

Code	Symbol	Meaning
0	.ENQBL	Queue the requests and block the process until all requested locks are acquired. The error return is taken only if the call is not correctly specified.
1	.ENQAA	Queue the requests and acquire the locks only if all requested resources are immediately available. No requests are queued and the error return is taken if any one of the resources is not available.
2	.ENQSI	Queue the requests. If all requested resources are immediately available, this function is identical to the .ENQBL function. If all resources are not immediately available, the request is queued and the error return (ENQX6:) is taken. A software interrupt will occur when all requested resources have been given to the process.
3	.ENQMA	Modify the access of a previously queued request. (Refer to EN%SHR below.) The access of each lock in this request is compared with the access of each lock in the previously queued request. If the two accesses are the same, no action is taken. If the access in this request is shared and the access in the previous request is exclusive, the successful return is taken. If the access in this request is exclusive and the access in the previous request is shared, the error return is taken unless this process is the only user of the lock. In this case, the successful return is taken. The error return is also taken if 1. Any one of the specified locks does not have a pending request. 2. Any one of the specified locks is a pooled resource. Each lock specified is checked, and the access is changed for all locks that were given correctly. If the error return is taken, the user must execute the ENQC call to determine the current state of each lock.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.ENQLN	length of the header and the number of requested locks in the left half, and length of argument block in the right half.
1	.ENQID	software interrupt channel number in the left half, and the request ID in the right half.

2	.ENQLV	flags and level number in the left half, and JFN, -1, -2, or -3 in the right half. (See below.)
3	.ENQUC	pointer to a string or a 5B2+33-bit user code.
4	.ENQRS	number of resources in pool in the left half and number of resources requested in the right half, or 0 in the left half and a group number in the right half.
5	.ENQMS	address of a resource mask block.
	.	
	.	
	.	
n-4		flags and level number in the left half, and JFN, -1, -2, or -3 in the right half.
n-3		pointer to a string or a 5B2+33-bit user code.
n-2		number of resources in pool in the left half and number of resources requested in the right half, or 0 in the left half and a group number in the right half.
n-1		address of a resource mask block.

The argument block is divided into two logical sections: a header and individual requests for each desired lock. Words .ENQLN and .ENQID form the header. Word .ENQLV through word .ENQMS form the individual request and are repeated for each lock being requested. The words in the argument block are described in the following paragraphs.

.ENQLN

The length of the header (.ENHLN) is contained in bits 0 through 5. Currently, the length of the header is two words. (Note that a given length of zero or one is assumed to be equal to a length of two.) The number of locks being requested (.ENNLK) is contained in bits 6 through 17, and the length of the argument block (.ENALN) is contained in bits 18 through 35.

.ENQID

The software interrupt channel specifies the number of the channel on which to generate an interrupt with the .ENQSI function. The request ID is an 18-bit user-generated value used to identify the particular resource. This ID currently is not used by the system but is stored for future expansion of the facility.

.ENQLV

The following flags are defined:

B0(EN%SHR) Access to this resource is to be shared. If this bit is not set, access to the resource is to be exclusive.

B1(EN%BLN) Ignore the level number associated with this resource. Sequencing errors in level numbers will not be considered fatal, and execution of the call will continue. If a sequencing error occurs, the successful return will be taken, and AC1 will contain an error code indicating the sequencing error that occurred.

B2(EN%NST) Allow ownership of this lock to be nested to any level within a process. This means that a process can request this resource again even though it already owns it. If the process has a request in the resource's queue or if the process already owns the lock, the ownership of the lock is nested to a depth one greater than the current depth. If the process does not have a request in the resource's queue, the setting of this bit has no effect, and the execution of the ENQ call continues. When a process has a nested lock, it must DEQ the resource as many times as it ENQed it before the resource becomes available to other processes.

B3(EN%LTL) Allow a long-term lock on this resource. This notifies the system that this resource will be locked and unlocked many times in a short period of time. Setting this bit permits a program to run faster if it is doing multiple locks and unlocks on the same resource because the argument block data is not deleted immediately from the ENQ/DEQ data base when a DEQ call is executed. Thus, the time required to re-create the data is reduced.

B9-B17 (EN%LVL) Level number associated with this resource.

The request is not queued and the error return is taken if EN%BLN is not set and

1. A resource with a level number less than or equal to the highest numbered resource requested so far is specified.

2. The level number of the current request does not match the level number supplied on previous requests for this resource.

The right half of .ENQLV specifies the type of access desired for the resource. If a JFN is given, the file associated with the JFN is subject to the standard access protection of the system. If -1 is given, the resource can be accessed only by processes of the job. If -2 is given, the resource can be accessed by any job on the system. (The process must have ENQ capability enabled to specify -2.) If -3 is given, the resource can be accessed only by processes that have WHEEL or OPERATOR capability enabled.

.ENQUC

This word is either a byte pointer to a string or a 33-bit user code, either of which serves to uniquely identify the resource to all users. This quantity is the second part of the resource name. (JFN, -1, -2, or -3 is the first part of the resource name.) The system makes no association between these identifiers and any physical resource.

The string can be comprised of bytes of any size from 1 to 36 bits in length and is terminated by a null byte. The byte size desired is specified by the pointer to the string. The maximum length of the string (including the terminating null byte) is 50 words long.

.ENQRS

This word is used to allocate multiple resources from a pool of identical resources. The total number of resources in the pool is a parameter agreed upon by all users. All requests for the same pooled resource must agree with the original count or the error return is taken. The number of resources being requested from the pool must be greater than zero if a pool exists and must be smaller than or equal to the number in the pool. If the left half of this word is zero, the system assumes only one resource of the specific type exists. In this case, if the right half of this word is positive, it is interpreted as the number of the group of users who can simultaneously access the resource.

.ENQMS

This word is used to obtain a single lock representing many specific resources. For example, a lock can be obtained on a particular data base, and the specific resources requested can be individual records in that data base.

This word contains an address of a mask block, where each bit in the mask represents a specific resource of the lock. The set of resources comprising the lock is a parameter agreed upon by all users.

A process can obtain exclusive access to all or some of the specific resources comprising the lock. When a process requires exclusive access to all of the resources, it executes an ENQ call (for exclusive access) and does not specify a mask block. A successful return will be given if there are no other processes that have issued an ENQ call for that lock. Otherwise, the process blocks until the requested resources are available.

When a process requires exclusive access to some of the specific resources comprising the lock, it sets up the mask block and sets the bits corresponding to the specific resources it wants to lock. The process then executes an ENQ call for exclusive access. On successful execution of the ENQ call, the process has an exclusive lock for the resources represented by the bits on in the mask. The process blocks if another process owns an exclusive lock on the resource and that process' ENQ call did not specify a mask block.

Once a mask block has been set up for a set of specific resources, subsequent requests for a different set of resources will be honored. The set of resources being requested is considered different if the bits on in one process' mask block are not any of the same bits on in another process' mask block. When a subsequent request is given for resources that are currently locked by a process, the process with the request blocked until the last of the currently locked resources is dequeued by the owner of the lock.

A process can dequeue all or part of the original ENQ call request. When a DEQ call is executed, the bits on in the mask block of the DEQ call are compared with the bits on in the original ENQ call. The resources not being dequeued remain locked and must be dequeued by a subsequent DEQ call. This action allows a process to lock a number of resources all at once and then to release individual resources as it finishes with them. However, a process cannot execute subsequent ENQ calls to request additional resources from those requested in its original ENQ call.

The first word of the mask block contains a count in the right half of the number of words in the block, including this count word. Remaining words contain the 36 mask bits. The maximum length of the mask block is 16 words. All requests for the resources associated with the mask block must specify the same length for the block or an error return is taken. Also, when a mask block is specified, the ENQ call must request exclusive access to the resource and the left half of word .ENQRS of the lock request must be zero.

ENQ ERROR MNEMONICS:

ENQX1: invalid function

ENQX2: level number too small

ENQX3: request and lock level numbers do not match

ENQX4: number of pool and lock resources do not match

ENQX5: lock already requested

ENQX6: requested locks are not all locked

ENQX8: invalid access change requested

ENQX9: invalid number of blocks specified

ENQX10: invalid argument block length

ENQX11: invalid software interrupt channel number

ENQX12: invalid number of resources requested

ENQX13: indirect or indexed byte pointer not allowed

ENQX14: invalid byte size

ENQX15: ENQ/DEQ capability required

ENQX16: WHEEL or OPERATOR capability required

ENQX17: invalid JFN

ENQX18: quota exceeded

ENQX19: string too long

ENQX20: locked JFN cannot be closed

ENQX22: invalid mask block length

ENQX23: mismatched mask block lengths

DESX8: file is not on disk

ENQC JSYS 515

Returns the current status of the given resource and obtains information about the state of the queues. This monitor call also allows privileged processes to manipulate access rights to the queues and to perform other utility functions on the queue structure.

Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

The ENQC monitor call has two calling sequences, depending on whether the process is obtaining status information or is modifying the queue structure.

Obtaining Status Information

ACCEPTS IN	AC1:	function code (.ENQCS)
	AC2:	address of argument block
	AC3:	address of block in which to place status
RETURNS	+1:	failure, error code in AC1
	+2:	success

The function .ENQCS returns the status of the specified resources.

The argument block is identical in format to the ENQ and DEQ argument blocks. (Refer to the ENQ monitor call description.)

The status block has a 3-word entry for each resource specified in the argument block. This entry reflects the current status of the resource and has the following format:

```
 0                        17 18                       35
!=======================================================!
!       flag bits indicating status of resource         !
!=======================================================!
!                    36-bit time stamp                  !
!=======================================================!
! # of processes with lock  !        request ID         !
!=======================================================!
```

The following flag bits are currently defined.

B0(EN%QCE) An error has occurred in the corresponding resource request and bits 18-35 contain an appropriate error code.

B1(EN%QCO) This process owns the lock.

B2(EN%QCQ) This process is in the queue waiting for this resource. This bit is set if B1(EN%QCO) is set because a request remains in the queue until a DEQ call is given.

B3(EN%QCX) The lock has been allocated for exclusive access.

B4(EN%QCB)	This process is in the queue waiting for exclusive access to the resource. This bit is off if B2(EN%QCQ) is off.
B9-B17 (EN%LVL)	The level number of the resource.
B18-B35 (EN%JOB)	Job number of the owner of the lock. For locks with shared access, this value will be the job number of one of the sharers. However, this value will be the current job's number if the current job is one of the sharers. If the lock is not owned, the value is -1. If B0(EN%QCE) is on, this field contains the appropriate error code.

The time stamp indicates the last time a process was given access to the resource. The time is in the universal date-time standard. If no process currently has access to the resource, the word is zero.

The number returned in the left half of the third word indicates the number of processes that currently have the resource locked for either exclusive access or shared access.

The request ID is either the request ID of the current process if that process is in the queue or the request ID of the owner of the lock.

Modifying the Queue Structure

ACCEPTS IN AC1: function code

AC2: address of argument block

RETURNS +1: failure, error code in AC1

+2: success

The available functions, along with their argument block formats, are as follows:

Function	Argument Block	Meaning
.ENQCG	One word containing a job number in the right half. The left half is ignored.	Return the ENQ/DEQ quota for the specified job. The quota is returned in AC1.
.ENQCC	One word containing the new quota in the left half and a job number in the right half.	Change the ENQ/DEQ quota for the specified job. The process executing the call must have WHEEL capability enabled or an error code is returned.
.ENQCD	A block of n words. The first word is the length of the block (n). Remaining words contain the returned data. (See below.)	Dump the ENQ/DEQ locks and queue entries into the argument block. The process executing the call must have WHEEL capability enabled or an error code is returned.

The data returned in the argument block is data concerning both the ENQ/DEQ locks and the queues. The data concerning the locks is in a 4-word block of the following format:

```
          0           8 9          17 18                        35
          !=======================================================!
.ENQDF    !    flags    !level number ! OFN, 40000+job#, -2, or -3!
          !=======================================================!
.ENQDR    !  total resources in pool  ! # of resources remaining  !
          !=======================================================!
.ENQDT    !            time stamp of last request locked          !
          !=======================================================!
.ENQDC    !       user code of lock or beginning of string        !
          !=======================================================!
```

If there are no pooled resources, word .ENQDR has the format:

```
          0                        17 18                        35
          !=======================================================!
.ENQDR    !             0             !      group number         !
          !=======================================================!
```

The data concerning the queues is in a 2-word block of the following format:

```
          0           8 9          17 18                        35
          !=======================================================!
.ENQDF    !    flags    !software chan! job # creator queue entry !
          !=======================================================!
.ENQDI    !group # or number requested!        request ID         !
          !=======================================================!
```

The flags returned in the first word of each block are as follows:

B0(EN%QCL) This block concerns data about the locks. If this bit is off, the block concerns data about the queues.

B1(EN%QCO) This process owns the lock.

B2(EN%QCT) This lock contains a text string.

B3(EN%QCX) This lock is for exclusive access.

B4(EN%QCB) This process is blocked until exclusive access is available.

ENQC ERROR MNEMONICS:

ENQX1: invalid function

ENQX2: level number too small

ENQX3: request and lock level numbers do not match

ENQX4: number of pool and lock resources do not match

ENQX5: lock already requested

ENQX6:	requested locks are not all locked
ENQX7:	no ENQ on this lock
ENQX8:	invalid access change requested
ENQX9:	invalid number of blocks specified
ENQX10:	invalid argument block length
ENQX11:	invalid software interrupt channel number
ENQX12:	invalid number of resources requested
ENQX13:	indirect or indexed byte pointer not allowed
ENQX14:	invalid byte size
ENQX15:	ENQ/DEQ capability required
ENQX16:	WHEEL or OPERATOR capability required
ENQX17:	invalid JFN
ENQX18:	quota exceeded
ENQX19:	string too long
ENQX20:	locked JFN cannot be closed
ENQX21:	job is not logged in
DESX8:	file is not on disk

EPCAP JSYS 151

Enables the capabilities for the specified process. (Refer to Section 2.6.1 for a description of the capability word.)

ACCEPTS IN	AC1:	process handle
	AC2:	capabilities possible for the specified process
	AC3:	capabilities to enable for the specified process
RETURNS	+1:	always

The capabilities in bits 0-8 and bits 18-35 of AC2 are matched with the corresponding capabilities of the process executing the call. If the executing process does not have the capability available, it cannot be enabled for the specified process (i.e., an AND operation is performed).

The contents of AC2 are ignored if the process handle in AC1 is for the current process.

The RPCAP monitor call can be used to obtain the capabilities of a process.

Generates an illegal instruction interrupt on error conditions below.

EPCAP ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

ERSTR JSYS 11

Translates a TOPS-20 error number to its corresponding text string and writes the string to the specified destination. This error number is the one returned in an AC (usually in AC1) on a JSYS error and is associated with a unique error mnemonic and text string. The error numbers begin at 600010 and are defined in the system file MONSYM.MAC. (Refer to Appendix A for the list of error numbers, mnemonics, and text strings.)

ACCEPTS IN AC1: destination designator

AC2: LH: process handle
RH: error number, or -1 for the most recent error in the specified process

AC3: LH: a negative count of the maximum number of bytes in the string to be transferred, or 0 for no limit
RH: 0

RETURNS +1: failure, undefined error number

+2: failure, string size out of bounds or invalid destination designator

+3: success

Generates an illegal instruction interrupt on error conditions below.

ERSTR ERROR MNEMONICS:

DESX1: invalid source/destination designator

FRKHX1: invalid process handle

IOX11: quota exceeded or disk full

ESOUT JSYS 313

Outputs an error string. This monitor call is used for reporting an error in the input from the primary input stream in order to cause re-synchronization of the input transaction. This mechanisim is convenient for communication with a user who made a typing error and may have continued to type ahead. It also standardizes the format of error messages.

ACCEPTS IN AC1: pointer to a string in the caller's address space. The string is terminated with a null character.

RETURNS +1: always, updated string pointer in AC1

The ESOUT call waits for the primary output buffer to empty and then outputs a carriage return, line feed, and question mark to the primary output designator. Next it clears the primary input buffer and outputs the error string to the primary output designator.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

FFFFP JSYS 31

Finds the first free page in the specified file. A free page is one that is marked as not being in use. The FFFFP call is useful for finding a nonused page in a file before a PMAP call is executed that writes into that page.

ACCEPTS IN AC1: JFN

RETURNS +1: always, with the JFN in the left half of AC1 and the page number in the right half of AC1, or -1 if there is no free page.

Generates an illegal instruction interrupt on error conditions below.

FFFFP ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: illegal use of terminal designator or string pointer

DESX5: file is not open

FFORK JSYS 154

Freezes one or more processes.

ACCEPTS IN AC1: process handle

RETURNS +1: always

This suspends the processes (as soon as they are stoppable from the monitor's point of view) in such a way that they can be continued at the place they were suspended. However, they do not have to be continued; they could be killed.

The FFORK call is a no-op if the referenced process is already frozen.

The RFORK monitor call can be used to resume one or more processes.

Generates an illegal instruction interrupt on error conditions below.

FFORK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

FFUFP JSYS 211

Finds the first used page of the file at or beyond the specified page number.

ACCEPTS IN AC1: JFN in the left half, and the starting page number in the right half

RETURNS +1: failure, error code in AC1

+2: success, page number in the right half of AC1. The left half of AC1 is unchanged.

FFUFP ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: illegal use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

FFUFX1: file is not open

FFUFX2: file is not on multiple-directory device

FFUFX3: no used page found

FLIN JSYS 232

Inputs a floating-point number from the specified source. This call ignores leading spaces and terminates on the first character that cannot be part of a floating point number. If that character is a carriage return followed by a line feed, the line feed is also input.

ACCEPTS IN AC1: source designator

RETURNS +1: failure, error code in AC3 and updated string pointer in AC1, if pertinent

+2: success, single-precision, floating-point number in AC2 and updated string pointer in AC1, if pertinent

FLIN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

FLINX1: first character is not blank or numeric

FLINX2: number too small

FLINX3: number too large

FLINX4: invalid format

FLOUT JSYS 233

Outputs a floating-point number to the specified destination.

ACCEPTS IN AC1: destination designator

AC2: normalized, single-precision, floating-point number

AC3: format control word. (Refer to Section 2.8.1.2.)

RETURNS +1: failure, error code in AC3 and updated string pointer in AC1, if pertinent

+2: success, updated string pointer in AC1, if pertinent

FLOUT ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: file is not open

FLOTX1: column overflow in field 1 or 2

FLOTX2: column overflow in field 3

FLOTX3: invalid format specified

IOX11: quota exceeded or disk full

GACCT JSYS 546

Returns the current account for the specified job.

ACCEPTS IN AC1: job number, or -1 for current job

AC2: pointer to string where alphanumeric account designator (if any) is to be stored

RETURNS +1: always, with updated pointer to account string in AC2

The GACCT monitor call requires the process to have Confidential Information Access, WHEEL, or OPERATOR capability enabled if the specified job number is not for the current job.

The CACCT monitor call can be used to change the account for the current job.

Generates an illegal instruction interrupt on error conditions below.

GACCT ERROR MNEMONICS:

GACCX1: invalid job number

GACCX2: no such job

GACCX3: Confidential Information Access capability required

GACTF JSYS 37

Returns the account designator to which the specified file is being charged.

ACCEPTS IN AC1: JFN

AC2: pointer to string in caller's address space where account string (if any) is to be stored

RETURNS +1: failure, error code in AC1

+2: success, account string returned, updated string pointer in AC2

+3: success, 5B2+account number returned in AC2

The SACTF monitor call can be used to set the account designator to which the file is to be charged.

GACTF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

GACTX1: file is not on multiple-directory device

GACTX2: file expunged

GACTX3: internal format of directory is incorrect

GCVEC JSYS 300

Returns the entry vector and the UUO locations for the compatibility package.

ACCEPTS IN AC1: process handle

RETURNS +1: always, entry vector length in the left half and entry vector address in the right half of AC2, and UUO location in the left half and PC location in the right half of AC3.

If use of the compatibility package has been disabled, AC2 contains -1 on return. If the compatibility package is not available, AC2 and AC3 contain 0 on return.

The SCVEC monitor call can be used to set the entry vector for the compatibility package.

GCVEC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

GDSKC JSYS 214

Returns information on the given structure's disk usage and availability. This call is useful in determining storage usage.

ACCEPTS IN AC1: device designator, must be a designator for a structure. If the designator for DSK: is given, the connected structure is assumed.

RETURNS +1: always, number of pages in use in AC1, and number of pages not in use in AC2.

GDSKC ERROR MNEMONICS:

DEVX1: invalid device designator

GDSTS JSYS 145

Returns the status of a device. (Refer to Section 2.4 for the descriptions of the status bits.) This call requires that the device be opened.

ACCEPTS IN AC1: JFN

RETURNS +1: always, device-dependent status bits in AC2, and device-dependent information in AC3. For magnetic tape, AC3 contains the positive count of number of hardware bytes actually transferred in the left half and 0 in the right half. For the line printer, AC3 contains the last value of the page counter register, or -1 if there is no page counter register.

The GDSTS call is a no-op for devices without device-dependent status bits.

The SDSTS monitor call can be used to set the status bits for a particular device.

Generates an illegal instruction interrupt on error conditions below.

GDSTS ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

GDVEC JSYS 542

Returns the entry vector for the Record Management System (RMS).

ACCEPTS IN AC1: process handle

RETURNS +1: always, entry vector length in the left half and the entry vector address in the right half of AC2.

The SDVEC monitor call can be used to set the entry vector for RMS.

Generates an illegal instruction interrupt on error conditions below.

GDVEC ERROR MNEMONICS:

ILINS5: RMS facility is not available

GET JSYS 200

Gets a save file, copying or mapping it into the process as appropriate, and updates the process' entry vector from the file. This call can be executed for either sharable or nonsharable save files that were created with the SSAVE or SAVE monitor call, respectively. The file must not be open.

ACCEPTS IN AC1: process handle in left half, and flag bits and a JFN in right half.

AC2: lowest page number in left half, and highest page number in right half. This controls the parts of memory that are loaded when GT%ADR is on.

RETURNS +1: always

The defined flag bits in AC1 are as follows:

Bit	Symbol	Meaning
19	GT%ADR	Use the memory address limits given in AC2. If this bit is off, all existing pages of the file (according to its directory) are mapped.
20	GT%PRL	Preload the pages being mapped (i.e., move the pages immediately.) If this bit is off, the pages are read in from the disk when they are referenced.
21	GT%NOV	Do not overlay existing pages and do return an error. If this bit is off, existing pages will be overlaid.
22	GT%FL2	Read additional flag bits specified in AC3 (reserved for future development).

The JFN is given in bits 24 through 35 of AC1.

When the GET call is executed for a sharable save file, pages from the file are mapped into pages in the process, and the previous contents of the process' page are overwritten. If the file contains data for only a portion of the process' page, the remainder of the page is zeroed. Pages of the process not used by the file are unchanged.

When the GET call is executed for a nonsharable save file, individual words of the file are written into the process. Since these files usually do not have words containing all zeros, a GET call executed for a nonsharable file never clears memory.

The GET call never loads the accumulators.

Can cause several software interrupts or process terminations on certain file conditions.

Generates an illegal instruction interrupt on error conditions below.

GET ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

GETX1: invalid save file format

GETX2: system Special Pages Table full

GETX3: illegal to overlay existing pages

SSAVX1: illegal to save files on this device

OPNX2: file does not exist

All file errors can occur.

GETAB JSYS 10

Returns a word from the specified system table. (Refer to Section 2.3.2.)

ACCEPTS IN AC1: index into table in the left half, and table number in the right half

RETURNS +1: failure, error code in AC1

+2: success, 36-bit word from the specified table in AC1

If -1 is given as the index, this call returns the negative of the length of the specified table.

The table number can be obtained with the SYSGT call. However, the recommended procedure is to use the symbol definition from the MONSYM file for the table number. (Refer to Appendix A for the system table definitions.)

The GETAB monitor call requires the process to have GETAB capability available, but not enabled (SC%GTB in the process capability word).

GETAB ERROR MNEMONICS:

GTABX1: invalid table number

GTABX2: invalid table index

GTABX3: GETAB privileges required

GETER JSYS 12

Returns the most recent error condition encountered in a process. The most recent error is always saved in the Process Storage Block.

ACCEPTS IN AC1: process handle

RETURNS +1: always, process handle in left half of AC2 and most recent error condition in right half of AC2.

The SETER monitor call can be used to set the last error condition encountered in a process.

GETER ERROR MNEMONICS:

LSTRX1: process has not encountered any errors

GETJI JSYS 507

Returns information for the specified job.

ACCEPTS IN AC1: job number, or -1 for current job, or 400000+TTY number

AC2: negative of the length of the block in which to store the information in the left half, and the beginning address of the block in the right half

AC3: word number (offset) of first entry desired from job information table

RETURNS +1: failure, error code in AC1

+2: success, with updated pointer in AC2 and requested entries stored in specified block

When a terminal designator is given in AC1, the information returned is for the job running on that terminal.

The system begins copying the entries from the job information table, starting with the offset given in AC3, into the address specified in the right half of AC2. The number of entries copied is minus the number given in the left half of AC2 or is the number remaining in the table, whichever is smaller. Because AC2 is updated on a successful return, it cannot be used for the returned data.

The format of the job information table is as follows:

Word	Symbol	Meaning
0	.JIJNO	Job number
1	.JITNO	Job's terminal number (-1 means the job is detached)
2	.JIUNO	Job's user number
3	.JIDNO	Job's connected directory number
4	.JISNM	Subsystem name
5	.JIPNM	Program name
6	.JIRT	Run time
7	.JICPJ	Controlling PTY job number (-1 means the job is not controlled by a PTY)
10	.JIRTL	Run time limit (as set by the TIMER call)
11	.JIBAT	Job is controlled by Batch, if -1 (as set by the MTOPR call)
12	.JIDEN	Default for magnetic tape density (as set by the SETJB call)
13	.JIPAR	Default for magnetic tape parity (as set by the SETJB call)
14	.JIDM	Default for magnetic tape data mode (as set by the SETJB call)
15	.JIRS	Default for magnetic tape record size (as set by the SETJB call)
16	.JIDFS	Deferred spooling in effect, if 1 (as set by the SETJB call)
17	.JILNO	Job's logged-in directory number
20	.JISRM	Pointer to job's session remark
21	.JILLN	Job's last login date and time

GETJI ERROR MNEMONICS:

GTJIX1: invalid index

GTJIX2: invalid terminal line number

GTJIX3: invalid job number

GTJIX4: no such job

GETNM JSYS 177

Returns the name of the program currently being used by the job. This name will have been declared previously with the SETNM or SETSN monitor call.

RETURNS +1: always, SIXBIT name of program in AC1

GEVEC JSYS 205

Returns the entry vector of the specified process. (Refer to Section 2.7.3.)

ACCEPTS IN AC1: process handle

RETURNS +1: always, specified process' entry vector word in AC2

The SEVEC monitor call can be used to set the process' entry vector.

Generates an illegal instruction interrupt on error conditions below.

GEVEC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

GFRKH JSYS 164

Gets a handle on a process that currently is not known to the caller but is known to another process. The handle returned can then be used by the caller to refer to the desired process.

ACCEPTS IN AC1: process handle of the process that knows about the desired process and that currently has a handle on it.

AC2: process handle used by the process in AC1 to refer to the desired process. This handle must be a relative handle (i.e., in the range 400000 to 400777) and must refer to an existing process.

RETURNS +1: failure, with error code in AC1.

+2: success, with a handle in AC1 that is usable by the caller to refer to the desired process. This handle is not the same as the one given in AC2 (i.e., is different from the one used by the process in AC1 to refer to the desired process).

Generates an illegal instruction interrupt on error conditions below.

GFRKH ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

FRKHX6: all relative process handles in use

GFRKX1: invalid process handle

GFRKS JSYS 166

Returns the process structure of the current job from a given process downward.

ACCEPTS IN AC1: process handle of the starting point

AC2: B0(GF%GFH) return relative process handles for each process

B1(GF%GFS) return status for each process

AC3: negative of the word count of the block in which to store the structure in the left half, and the address of the first word of the block in the right half

RETURNS +1: failure, error code in AC1

+2: success, all process handles are returned

The handle of the current process is always returned as .FHSLF regardless of the setting of GF%GFH. Any user can specify a process handle of .FHTOP (i.e., start with the top level process). However, the user must have WHEEL or OPERATOR capability enabled to specify .FHTOP and set GF%GFH; otherwise, the setting of GF%GFH is ignored.

Table format

```
              *********************************************
              *                    *                      *
3 words       *     parallel       *      inferior        *
per entry     *     pointer        *      pointer         *
              *                    *                      *
              *********************************************
              *                    *                      *
              *     superior       * process handle       *
              *     pointer        * or 0 if GF%GFH       *
              *                    * was off, or when no  *
              *                    * more process handles *
              *                    * are left for the     *
              *                    * process              *
              *                    *                      *
              *********************************************
              *                                           *
This word is  *              status word                  *
-1 if GF%GFS  *                                           *
is off.       *                                           *
              *********************************************
```

NOTE

Pointers in table are memory addresses of other table entries, or 0 if no such structure.

The execution of the GFRKS call will be terminated before the entire structure has been returned if either all handles are in use or the block in which to store the structure is too small. If this happens, as much of the structure as possible is returned before an error code is generated.

Generates an illegal instruction interrupt on error conditions below.

GFRKS ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

FRKHX6: all relative process handles in use

GFKSX1: area too small to hold process structure

GFUST JSYS 550

Returns the name of either the author of the file or the user who last wrote to the file.

ACCEPTS IN AC1: function code in the left half, and JFN of the file in the right half

AC2: pointer to the string in which to store the name

RETURNS +1: always, with an updated string pointer in AC2

The defined functions are as follows:

Code	Symbol	Meaning
0	.GFAUT	Return the name of the author of the file.
1	.GFLWR	Return the name of the user who last wrote to the file.

The SFUST monitor call can be used to set the name of either the author of the file or the user who last wrote to the file.

Generates an illegal instruction interrupt on error conditions below.

GFUST ERROR MNEMONICS:

GFUSX1: invalid function

GFUSX2: insufficient system resources

GFUSX3: file expunged

GFUSX4: internal format of directory is incorrect

DESX1: invalid source/destination designator

DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX5:	file is not open
DESX7:	JFN cannot refer to output wildcard designators
DESX8:	file is not on disk
DESX10:	structure is dismounted
DELFX6:	internal format of directory is incorrect
DIRX2:	insufficient system resources
DIRX3:	internal format of directory is incorrect

GJINF JSYS 13

Returns information pertaining to the current job.

RETURNS +1: always, with

AC1 containing the user number under which the job is running.

AC2 containing the directory number to which the job is connected.

AC3 containing the job number.

AC4 containing the terminal number attached to the job, or -1 if no terminal is attached to job.

GNJFN JSYS 17

Assigns the JFN to the next file in a group of files that have been specified with wildcard characters. The next file in the group is determined by searching the directory in the order described in Section 2.2.3 (i.e., in the internal directory order). The flags returned from the GTJFN call are given to the GNJFN call as an argument to indicate the fields of the file specification that contain wildcard characters.

ACCEPTS IN AC1: indexable file handle returned by GTJFN (i.e., flags returned by GTJFN in the left half and the JFN in the right half)

RETURNS +1: failure, including no more files in the group. JFN is released if there are no more files in the group. This return occurs on the first call to GNJFN if no flags indicating wildcard fields are on in the left half of AC1.

+2: success, same JFN is assigned to the next file in the group. The following flags are set (if appropriate) in the left half of AC1:

B13	GN%STR	structure changed
B14	GN%DIR	directory changed
B15	GN%NAM	name changed
B16	GN%EXT	file type changed

The GNJFN call uses the flags returned in the left half of AC1 on a GTJFN call to determine the fields containing wildcards and the default generation number. Note that the GNJFN call returns a different set of flags in the left half of AC1 than the GTJFN call returns. Because all calls to GNJFN should use the flags originally returned by GTJFN, programs must save the returned GTJFN flags for use in the GNJFN call.

The file currently associated with the JFN must be closed when the GNJFN call is executed.

GNJFN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

GNJFX1: no more files in this specification

OPNX1: file is already open

GPJFN JSYS 206

Returns the primary JFNs of the specified process.

ACCEPTS IN AC1: process handle

RETURNS +1: always, primary input JFN in the left half of AC2, and the primary output JFN in the right half of AC2.

The SPJFN monitor call can be used to set the primary JFNs. If this call has not been given, the GPJFN call returns -1 in AC2.

Generates an illegal instruction interrupt on error conditions below.

GPJFN ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

GTAD JSYS 227

Returns the current date in the internal system format. (Refer to Section 2.8.2.)

RETURNS +1: always, day in the left half of AC1, and fraction of day in right half of AC1

If the system does not have the current date set, AC1 contains -1.

The STAD monitor call can be used to set the system's date.

GTDAL JSYS 305

Returns the disk allocation for the specified directory.

ACCEPTS IN AC1: directory number (-1 indicates the connected directory)

RETURNS +1: always, with

AC1 containing the working disk storage limit (logged-in quota) for the directory.

AC2 containing the number of pages being used.

AC3 containing the permanent disk storage limit (logged-out quota) for the directory.

Generates an illegal instruction interrupt on error conditions below.

GTDAL ERROR MNEMONICS:

DIRX1: invalid directory number

DELFX6: internal format of directory is incorrect

GTDIR JSYS 241

Returns information about the given directory.

ACCEPTS IN AC1: directory number

AC2: address of argument block in caller's address space in which to store the directory information

AC3: pointer to string in which to store the password

RETURNS +1: always, updated string pointer in AC3

The argument block returned to the caller is compatible with the one given on the CRDIR call. Word 0(.CDLEN) contains the length of the argument block in which to store the directory information being returned. If this word is zero, the length of the argument block is assumed to be 15 octal words long. The password of the directory is placed in the string indicated by AC3, and word 1(.CDPSW) of the returned argument block points to this string. Because the group list format includes a count word, the number of groups returned is one less than the count. The group list is terminated by a zero word.

If the given directory number is zero, the GTDIR monitor call returns the system default settings for the following directory parameters:

working disk storage quota (.CDLIQ)
permanent disk storage quota (.CDLOQ)
default file protection (.CDFPT)
default directory protection (.CDDPT)
default file retention count (.CDRET)
maximum number of subdirectories allowed (.CDSDQ)

Either one of the following conditions must be satisfied for the caller to obtain all information (including the password) about the given directory.

1. The caller has WHEEL or OPERATOR capability enabled.

2. The caller is connected to the directory that is immediately superior to the given directory.

To obtain all information other than the password of the given directory, the caller must have at least owner access to the directory. (Refer to Section 2.2.6 for a description of owner access.)

Generates an illegal instruction interrupt on error conditions below.

GTDIR ERROR MNEMONICS:

GTDIX1: WHEEL or OPERATOR capability required

GTDIX2: invalid directory number

GTFDB JSYS 63

Returns some or all of the file descriptor block for the specified file. (Refer to Section 2.2.8 for the format of this block.)

The GFUST monitor call must be used to obtain the name strings of the author of the file (.FBAUT) and of the user who last wrote the file (.FBLWR). The GACTF monitor call is used to obtain the account designator of the file (.FBACT).

ACCEPTS IN AC1: JFN

AC2: number of words to be read in the left half and the word number (offset) of the first entry desired from the file descriptor block in the right half.

AC3: address in caller's address space for storing the data returned

RETURNS +1: always

The program receives an error (GFDBX2) if it requests more words than there are words remaining in the FDB.

The CHFDB monitor call can be used to change most words in the file descriptor block for a specified file. Section 2.2.8 lists the words that can be changed with CHFDB. The SFUST monitor call is used to change the name strings of the author (.FBAUT) and the last writer (.FBLWR). The SACTF call is used to change the account designator of the file (.FBACT).

Generates an illegal instruction interrupt on error conditions below.

GTFDB ERROR MNEMONICS:

GFDBX1: invalid displacement

GFDBX2: invalid number of words

GFDBX3: list access required

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

GTJFN JSYS 20
SHORT FORM

Returns a JFN for the specified file. Accepts the specification for the file from a string in memory or from a file, but not from both. The string can represent the complete specification for the file:

dev:<directory>name.typ.gen;attributes

One or more fields of the specification can be defined by a logical name. (Refer to Section 2.2.2.) If any fields are omitted from the specification, the system will provide the values shown below.

device	connected structure
directory	connected directory
name	no default; this field must be specified
type	null
generation	highest existing number if the file is an input file. Next higher number if the file is an output file.
protection	protection as specified in the directory, or for new files, protection of the next lower generation.
account	account specified when user logged in, unless changed by the CACCT or SACTF call.

The JFNS monitor call can be used to obtain the file specification string associated with a given JFN.

ACCEPTS IN	AC1:	flag bits in the left half, and default generation number in the right half
	AC2:	source designator from which to obtain the file specification. (Refer to flag bit GJ%FNS for specific values.)
RETURNS	+1:	failure, error code in AC1
	+2:	success, flags in the left half of AC1, and the JFN assigned in the right half of AC1. (This word is called an indexable file handle and is given to the GNJFN call as an argument.) Updated string pointer in AC2, if pertinent.

All I/O errors can occur. These errors cause software interrupts or process terminations, and only a single return (+1) is given.

The flag bits that can be specified in AC1 are described as follows.

GTJFN Flag Bits

Bit	Symbol	Meaning
0	GJ%FOU	The file given is to be assigned the next higher generation number. This bit indicates that a new version of a file is to be created and is normally set if the file is for output use.
1	GJ%NEW	The file specification given must not refer to an existing file (i.e., the file must be a new file).
2	GJ%OLD	The file specification given must refer to an existing file (i.e., the file must be an old file).
3	GJ%MSG	One of the appropriate messages is to be printed after the file specification is obtained, if the system is performing recognition on the file specification and the user ends his input by typing an ESC. !NEW FILE! !NEW GENERATION! !OLD GENERATION! !OK! if GJ%CFM (bit 4) is off !CONFIRM! if GJ%CFM (bit 4) is on
4	GJ%CFM	Confirmation from the user will be required (if GJ%FNS is on) to verify that the file specification obtained is correct. (See below for the valid confirmation characters.)
5	GJ%TMP	The file specified is to be a temporary file.
6	GJ%NS	Only the first specification in a multiple logical name assignment is to be searched for the file (i.e., do not search beyond the first name in a multiple logical name assignment).
7	GJ%ACC	The JFN specified is not to be accessed by inferior processes in this job. However, another process can access the file by acquiring a different JFN. To prevent the file from being accessed by other processes, the user's program should set OF%RTD(B29) in the OPENF call.
8	GJ%DEL	Files marked as deleted are to be considered by the system when it is searching for a file to assign to the JFN.

9-10	GJ%JFN	These bits are off in the short form of the GTJFN call.
11	GJ%IFG	The file specification given is allowed to have one or more of its fields specified with a wildcard character (* or %). This bit is used to process a group of files and is generally used for input files. The monitor verifies that at least one value exists for each field that contains a wildcard and assigns the JFN to the first file in the group. The monitor also verifies that fields not containing wildcards represent a new or old file according to the setting of GJ%NEW and GJ%OLD. The GNJFN call can then be used to obtain the next file in the group. (Refer to Section 2.2.3 for more information on wildcard characters in file specifications.)
12	GJ%OFG	The JFN is to be associated with the given file specification string only and not to the actual file. The string may contain wildcard characters (* or %) in one or more of its fields. It is checked for correct punctuation between fields, but is not checked for the validity of xwy field. This bit allows a JFN to be associated with a file specification even if the file specification does not refer to an actual file. The JFN returned cannot be used to refer to an actual file (e.g., cannot be used in an OPENF call) but can be used to obtain the original input string (via JFNS). The fields in this string can then be used in a GTJFN-long form call as program defaults. However, if the original string contains the temporary file attribute (;T), this attribute is not "remembered" and thus is not returned on the JFNS call even though the bit indicating temporary status (JS%TMP) is set. All other fields (including the protection and account fields) can be returned by JFNS.

12	CJ%OFG (Cont.)	When both B11(GJ%IFG) and B12(GJ%OFG) are on, the GTJFN call parses the specification given, verifying the existence of each field. When a wildcard character appears in a field, the GTJFN call checks the remaining fields for correct punctuation and returns a JFN for the file specification string only. That is, once a wildcard character is seen, the action taken is identical to that taken when only B12(GJ%OFG) is set. If no wildcard character appears in the string, the action is the same as if both bits were off.
13	GJ%FLG	Flags are to be returned in the left half of AC1 on a successful return.
14	GJ%PHY	User logical names specified for the current job are to be ignored and the physical device is to be used.
15	GJ%XTN	This bit is off in the short form of the GTJFN call.
16	GJ%FNS	The contents of AC2 are to be interpreted as follows: 1. If this bit is on, AC2 contains an input JFN in the left half and an output JFN in the right half. The input JFN is used to obtain the file specification to be associated with the JFN. The output JFN is used to indicate the destination for printing the names of any fields being recognized. To omit either JFN, specify .NULIO (377777). 2. If this bit is off, AC2 contains a pointer to an ASCIZ string in memory that specifies the file to be associated with the JFN.
17	GJ%SHT	This bit must be on for the short form of the GTJFN call.
18-35		The generation number of the file. The following values are permitted; however, 0 is the normal case. 0(.GJDEF) to indicate that the next higher generation number of the file is to be used if GJ%FOU (bit 0) is on, or to indicate that the highest existing generation number of the file is to be used if GJ%FOU is off.

18-35 (Cont.)	-1(.GJNHG)	to indicate that the next higher generation number of the file is to be used if no generation number is supplied.
	-2(.GJLEG)	to indicate that the lowest existing generation number of the file is to be used if no generation number is supplied.
	-3(.GJALL)	to indicate that all generation numbers (*) of the file are to be used and that the JFN is to be assigned to the first file in the group, if no generation number is supplied. (Bit GJ%IFG must be set.)
	1-377777	to indicate that the specified generation number of the file is to be used if no generation number is supplied.

The GTJFN monitor call always reads the terminating character after the file specification string. (This character can be obtained by executing the BKJFN call followed by a BIN call.) The valid terminating characters are:

line feed	left parenthesis
CTRL/L	right parenthesis
CTRL/Z	plus sign
carriage return	comma
exclamation point	slash
double quotation marks	equals sign
number sign	at sign (@)
ampersand	space
single quotation mark	ESC

All of these characters except for ESC are also confirmation characters (refer to bit GJ%CFM above) and are called confirming terminators. If a confirming terminator is typed after the string, a confirmation message will not be typed to the user nor will the user be required to confirm the string obtained, regardless of the setting of GJ%MSG and GJ%CFM.

On a successful return, the following flags are returned in the left half of AC1 if flag bit GJ%IFG, GJ%OFG, or GJ%FLG was on in the call.

Bits Returned on Successful GTJFN Call

Bit	Symbol	Meaning
0	GJ%DEV	The device field of the file specification contained wildcard characters.
1	GJ%UNT	The unit field of the file specification contained wildcard characters. This bit will never be set because wildcard characters are not allowed in unit fields.
2	GJ%DIR	The directory field of the file specification contained wildcard characters.
3	GJ%NAM	The filename field of the file specification contained wildcard characters.
4	GJ%EXT	The file type field of the file specification contained wildcard characters.
5	GJ%VER	The generation number field of the file specification contained wildcard characters.
6	GJ%UHV	The file used has the highest generation number because a generation number of 0 was given in the call.
7	GJ%NHV	The file used has the next higher generation number because a generation number of 0 or -1 was given in the call.
8	GJ%ULV	The file used has the lowest generation number because a generation number of -2 was given in the call.
9	GJ%PRO	The protection field of the file specification was given.
10	GJ%ACT	The account field of the file specification was given.
11	GJ%TFS	The file specification is for a temporary file.
12	GJ%GND	Files marked for deletion will not be considered when assigning JFNs in subsequent calls. This bit is set if GJ%DEL was not set in the call.

GTJFN ERROR MNEMONICS:

GJFX1:	desired JFN invalid
GJFX2:	desired JFN not available
GJFX3:	no JFNs available
GJFX4:	invalid character in filename
GJFX5:	field cannot be longer than 39 characters
GJFX6:	device field not in a valid position
GJFX7:	directory field not in a valid position
GJFX8:	directory terminating delimiter is not preceded by a valid beginning delimiter
GJFX9:	more than one name field is not allowed
GJFX10:	generation number is not numeric
GJFX11:	more than one generation number field is not allowed
GJFX12:	more than one account field is not allowed
GJFX13:	more than one protection field is not allowed
GJFX14:	invalid protection
GJFX15:	invalid confirmation character
GJFX16:	no such device
GJFX17:	no such directory name
GJFX18:	no such filename
GJFX19:	no such file type
GJFX20:	no such generation number
GJFX21:	file was expunged
GJFX22:	insufficient system resources (Job Storage Block full)
GJFX23:	directory full
GJFX24:	file not found
GJFX27:	file already exists (new file required)
GJFX28:	device is not on-line
GJFX30:	account is not numeric
GJFX31:	invalid wildcard designator
GJFX32:	no files match this specification
GJFX33:	filename was not specified

GJFX34: invalid character "?" in file specification

GJFX35: directory access privileges required

GJFX36: internal format of directory is incorrect

GJFX37: input deleted

GJFX38: file not found because output-only device was specified

GJFX39: logical name loop detected

GJFX40: undefined attribute in file specification

GJFX41: file name must not exceed 6 characters

GJFX42: file type must not exceed 3 characters

GJFX43: more than one ;T specification is not allowed

GJFX44: account string does not match

GJFX45: illegal to request multiple specifications for the same attribute

GJFX46: attribute value is required

GJFX47: attribute does not take a value

GJFX48: GTJFN input buffer is empty

GJFX49: invalid attribute for this device

IOX11: quota exceeded or disk full

DESX9: invalid operation for this device

GTJFN JSYS 20
LONG FORM

Returns a JFN for the specified file. Accepts the specification for the file from both a string in memory and from a file. If both are given as arguments, the string is used first, and then the file is used if more fields are needed to complete the specification. This form also allows the program to specify nonstandard values to be used for omitted fields and to request the assignment of a specific JFN.

ACCEPTS IN AC1: 0 in the left half, and address of the beginning of the argument table in the caller's address space in the right half

AC2: pointer to ASCIZ file specification string in the caller's address space, or 0 if none

RETURNS +1: failure, error code in AC1

+2: success, flags in the left half of AC1, and the JFN assigned in the right half of AC1. (This word is called an indexable file handle and is given to the GNJFN call as an argument.) Updated string pointer in AC2, if pertinent.

All I/O errors can occur. These errors cause software interrupts or process terminations, and only a single return (+1) is given.

The format of the argument table specified by the right half of AC1 is described below. Words 0 through 10 (.GJGEN-.GJJFN) must be supplied in the long form of the GTJFN call. The remaining words are optional, and if they are supplied, B15(GJ%XTN) of word .GJGEN must be on.

Word	Symbol	Meaning
0	.GJGEN	Flag bits in the left half and generation number in the right half. (See below.)
1	.GJSRC	Input JFN in the left half and output JFN in the right half. To omit either JFN, specify .NULIO (377777).
2	.GJDEV	Pointer to ASCIZ string that specifies the default device to be used when none is given. If this word is 0, the user's connected structure will be used.
3	.GJDIR	Pointer to ASCIZ string that specifies the default directory to be used when none is given. If this word is 0, the user's connected directory will be used.
4	.GJNAM	Pointer to ASCIZ string that specifies the default filename to be used when none is given. If this word is 0, either the string or the input JFN must supply the filename.
5	.GJEXT	Pointer to ASCIZ string that specifies the default file type to be used when none is given. If this word is 0, the null file type will be used.
6	.GJPRO	Pointer to ASCIZ string that specifies the default protection to be used when none is given. If this word is 0, the default protection as specified in the directory or the protection of the next lower generation will be used.
7	.GJACT	Pointer to ASCIZ string that specifies the default account to be used when none is given. If this word is 0, the user's LOGIN account (unless changed) will be used.
10	.GJJFN	The JFN to associate with the file specification if flag GJ%JFN is set in word 0 (.GJGEN) of the argument block.

11	.GJF2	Extended argument block if B15(GJ%XTN) is on in the left half of .GJGEN. This word contains a second group of flags in the left half and the count of the number of words following this word in the argument block in the right half. The flags in the left half specify additional control over the GTJFN process. The following flags are defined: B0(G1%RND) Return to the caller if the filename buffer becomes empty, and the user attempts to delete a character. This can occur if the user, when giving the filename, types a CTRL/U or types a DELETE or CTRL/W and there are no more characters in the buffer. B2(G1%NLN) Filenames cannot be longer than 6 characters and file types cannot be longer than 3 characters. In addition, the generation number, temporary status, protection, and account fields cannot be specified in the string or the input data. B3(G1%RCM) Return the confirmation message to the caller by placing it in the destination buffer. B4(G1%RIE) Return to the caller if the input buffer becomes empty, and the user attempts to delete a character.
12	.GJCPP	Pointer to string where GTJFN is to store the exact copy of the user's typescript (destination string pointer). This string will contain logical names, if they were typed by the user, and will not contain the default fields unless they were generated through recognition. This string allows the caller to obtain a true copy of the user's typescript.
13	.GJCPC	Number of bytes available in the destination string pointed to by .GTCPP (word 12). If a pointer has been specified but this word is 0, the monitor assumes the string contains 130 bytes.
14	.GJRTY	Pointer to the buffer for text to be output when the user types a CTRL/R (i.e., pointer to the CTRL/R buffer). This pointer cannot be equal to the pointer given in AC2. (Refer to the TEXTI call for the definition of the CTRL/R buffer.)
15	.GJBFP	Pointer to the beginning of the destination buffer.
16	.GJATR	Pointer to the file specification attribute block. This word is reserved for future use.

The flag bits accepted in the left half of .GJGEN (word 0) of the argument block are basically the same as those accepted in the short form of the GTJFN call. The entire set of bits is listed below. (Refer to GTJFN - SHORT FORM for more detailed explanations of these bits.) The flags that are different in the two forms are GJ%JFN, GJ%XTN, GJ%FNS, and GJ%SHT.

Bit	Symbol	Meaning
0	GT%FOU	A new version of the file is to be created.
1	GJ%NEW	The file must not exist.
2	GJ%OLD	The file must exist.
3	GJ%MSG	A message is to be typed if an ESC is typed by the user when he terminates his input.
4	GJ%CFM	Confirmation from the user is required.
5	GJ%TMP	The file is temporary.
6	GJ%NS	Search only the first specification in a multiple logical name definition.
7	GJ%ACC	The JFN cannot be accessed by inferior processes.
8	GJ%DEL	The file deleted bit in the FDB is to be ignored.
9-10	GJ%JFN	The JFN supplied in .GJJFN (word 10) of the argument block is to be associated with the file specification. The value of this field is interpreted as follows:

Value	Meaning
0(.GJDNU)	Ignore the JFN supplied.
2(.GJERR)	Attempt to assign the JFN supplied and return an error if it is not available.
3(.GJALT)	Attempt to assign the JFN supplied and, if it is not available, assign an alternate.

Bit	Symbol	Meaning
11	GJ%IFG	The file specification is allowed to contain wildcard characters.
12	GJ%OFG	The JFN is to be associated with the file specification string and not the file itself.
13	GJ%FLG	Flags are to be returned in AC1 on successful completion of the call.
14	GJ%PHY	The physical device is to be used.
15	GJ%XTN	The argument block contains more than 10 (octal) words.
16	GJ%FNS	This bit is ignored for the long form of the GTJFN call.
17	GJ%SHT	This bit must be off for the long form of the GTJFN call.

The generation number given in the right half of .GJGEN (word 0) of the argument block can be one of the following:

0(.GJDEF)	to indicate that the next higher generation number is to be used if GJ%FOU is on, or to indicate that the highest existing generation number is to be used if GJ%FOU is off.
-1(.GJNHG)	to indicate that the next higher generation number is to be used if no generation number is supplied.
-2(.GJLEG)	to indicate that the lowest existing generation number is to be used if no generation number is supplied.
-3(.GJALL)	to indicate that all generation numbers are to be used and that the JFN is to be assigned to the first file in the group, if no generation number is supplied. (Bit GJ%IFG must be on.)
1-377777	to indicate that the specified number is to be used as the generation if no generation number is supplied.

On a successful return, flags are returned in the left half of AC1 if flag bit GJ%IFG, GJ%OFG, or GJ%FLG was on in the call. The flags returned are the same as those returned in the short form call of GTJFN.

Refer to the short form of the GTJFN call for the possible error mnemonics.

GTRPI JSYS 172

Returns the paging trap information for the specified process.

ACCEPTS IN AC1: process handle

RETURNS +1: always, with
- AC1 containing number of pager traps (i.e., the number of times a trap has occurred to the pager) for designated process since the process was started
- AC2 containing number of page faults (i.e., the number of times a trap has resulted in a page being swapped in) for designated process since the process was started
- AC3 containing time spent (in milliseconds) in page routines by designated process since the process was started

The number of pager traps will be greater than or equal to the number of page faults.

Generates an illegal instruction interrupt on error conditions below.

GTRPI ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

GTRPW JSYS 171

Returns the trap words. This monitor call allows a program to retrieve information about a previous read, write, or execute trap.

ACCEPTS IN AC1: process handle

RETURNS +1: always, trap status word from last memory trap in AC1, and last monitor call that had an error in AC2.

The following bits are defined in the status word:

B0(PF%USR)	page failure-user mode reference
B5(PF%WRT)	page failure-write reference
B14(TSW%RD)	trap status-read (always on)
B15(TSW%WT)	trap status-write (same setting as B5)
B16(TSW%EX)	trap status-execute (always on)
B17(TSW%MN)	trap status-monitor mode reference (complement of B0)
B18-B35	address of reference that caused the trap

This information allows a program to determine the exact cause of a memory trap and/or the effective virtual address that caused the trap. This information is sufficient to enable the program to continue, if desired, when the cause of the trap has been removed.

The contents of AC1 is 0 if there have been no memory traps.

Generates an illegal instruction interrupt on error conditions below.

GTRPW ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

GTSTS JSYS 24

Returns the status of a file associated with a JFN.

ACCEPTS IN AC1: JFN in the right half

RETURNS +1: always, status in AC2. If JFN is illegal in any way, B10 of AC2 will be 0.

JFN STATUS WORD

B0(GS%OPN)	file is open
B1(GS%RDF)	if file is open (i.e., bit 0 is on), it is open for read access
B2(GS%WRF)	if file is open, it is open for write access
B3(GS%XCF)	if file is open, it is open for execute access
B4(GS%RND)	if file is open, it is open for non-append access
B7(GS%LNG)	file is longer than 512 pages
B8(GS%EOF)	last read was past end of file
B9(GS%ERR)	file may be in error (i.e., a device or data error occurred)
B10(GS%NAM)	file specification is associated with this JFN
B11(GS%AST)	one or more fields of the file specification contain wildcard characters
B12(GS%ASG)	JFN is currently being assigned
B13(GS%HLT)	I/O errors are considered terminating conditions
B17(GS%FRK)	if file is open, it is open for restricted access
B32-B35 (GS%MOD)	data mode of the file. Refer to the OPENF monitor call description.

.GSNRM	normal data mode
.GSIMG	image (binary) mode
.GSDMP	dump data mode

If B0(GS%OPN) is not set on return, the file is not opened, and the settings of bits 1 through 4 are indeterminate.

The STSTS call can be used to set the status of a particular file.

GTTYP JSYS 303

Returns the terminal type number for the specified terminal line. (Refer to Section 2.4.3.4 for the terminal type numbers.)

ACCEPTS IN AC1: file designator (only terminal designators are legal)

RETURNS +1: always, terminal type number in AC2 and buffer allocation numbers (# of input buffers to be allocated in left half, and # of output buffers to be allocated in right half) in AC3. AC1 is unchanged.

The STTYP monitor call can be used to set the terminal type number for a specified line.

Generates an illegal instruction interrupt on error conditions below.

GTTYP ERROR MNEMONICS:

DESX1: invalid source/destination designator

TTYX01: line is not active

HALTF JSYS 170

Halts the current process and any inferior processes of the current process. Sets the process' PC to the next instruction after the call and saves it in the Process Storage Block (PSB) in case the process is continued. The user can continue the process by typing the CONTINUE command, which causes the process to start at the next instruction.

Sets bits 1-17(RF%STS) in the status word for this process to 2(.RFVPT). Refer to the RFSTS monitor call for the format of the status word.

If the top level process executes a HALTF call and does not have WHEEL or OPERATOR capability enabled, the job is logged out.

HFORK JSYS 162

Halts one or more inferior processes. (Refer to the HALTF monitor call description to halt the current process.)

ACCEPTS IN AC1: process handle (inferior processes only)

RETURNS +1: always

Sets bits 1-17(RF%STS) in the status word(s) for addressed process(s) to 2(.RFVPT). Refer to the RFSTS monitor call for the format of the status word.

Generates an illegal instruction interrupt on error conditions below.

HFORK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

HFRHX1: illegal to halt self with HFORK

HPTIM JSYS 501

Returns the value of one of the high precision system clocks. Although the main time base from interrupts generated by the internal system clock is in units of 1 millisecond, the clock provides a time base in units of 10 microseconds. The HPTIM monitor call provides access to the variables kept in these high precision units.

ACCEPTS IN AC1: number of the clock to read (see below)

RETURNS +1: failure, error code in AC1

+2: success, with AC1 containing the value of the specified clock

The numbers for currently-defined clocks are:

0	.HPELP	Elapsed time since system startup. (Refer to the TIME call for obtaining the time in milliseconds.)
1	.HPRNT	CPU runtime for this process. (Refer to the RUNTM call for obtaining the time in milliseconds.)

HPTIM ERROR MNEMONICS:

HPTX1: undefined clock number

HSYS JSYS 307

Initiates an orderly shutdown of the timesharing operation of the system. This call causes periodic notices of the impending shutdown to be issued to all terminals. It also causes any jobs still logged in at the designated shutdown to be logged out.

ACCEPTS IN AC1: shutdown time with the date and time in the internal format. (Refer to Section 2.8.2.)

AC2: date and time in internal format when system operation will resume (or 0 if unknown). Used for advisory messages only.

RETURNS +1: failure, error code in AC1

+2: success, shutdown procedure initiated

The shutdown notice is issued immediately to all terminals if the shutdown time is within two hours. The notice is also sent two hours, one hour, 30 minutes, 10 minutes, 5 minutes, and one minute before the shutdown.

The HSYS monitor call requires the process to have MAINTENANCE, OPERATOR, or WHEEL capability enabled. The time when the system is expected to be placed back into operation is not used directly by the monitor. It is entered into a GTTAB table where it may be examined with the GETAB monitor call.

HSYS ERROR MNEMONICS:

CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required

TIMEX1: time cannot be greater than 24 hours

TIMEX2: downtime cannot be more than 7 days in the future

IDCNV JSYS 223

Converts separate numbers for the local year, month, day, and time into the internal date and time format. (Refer to Section 2.8.2 for more information on the internal format.)

ACCEPTS IN AC2: year in the left half, and numerical month (0=January) in the right half

AC3: day of the month (0=first day) in the left half, and 0 in the right half

AC4:

B0(IC%DSA)	apply daylight savings according to the setting of B1(IC%ADS). If B0 is off, daylight savings is applied only if appropriate for the date.
B1(IC%ADS)	apply daylight savings if B0(IC%DSA) is on.
B2(IC%UTZ)	use time zone in B12-B17. If this bit is off, the local time zone is used.
B3(IC%JUD)	interpret the number in the right half of AC2 as being in Julian day format (Jan 1 is day 1).
B12-B17 (IC%TMZ)	time zone to use if B2(IC%UTZ) is on. (Refer to Section 2.8.2 for the time zones.)
B18-B35 (IC%TIM)	local time in seconds since midnight.

RETURNS +1: failure, error code in AC1

+2: success, AC2 contains the internal date and time, and AC3 contains

B0 and B2	on for compatibility with the ODCNV call
B1(IC%ADS)	on if daylight savings was applied
B12-B17 (IC%TMZ)	time zone used

IDCNV ERROR MNEMONICS:

DATEX1:	year out of range
DATEX2:	month is not less than 12
DATEX3:	day of month too large
DATEX5:	date out of range
TIMEX1:	time cannot be greater than 24 hours
ZONEX1:	time zone out of range

IDTIM JSYS 221

Inputs the date and time and converts them to the internal date and time format. (Refer to Section 2.8.2.) The IDTIM monitor call does not permit either the date or the time to be entered separately and does not perform conversions for time zones other than the local one (unless the time zone is specified in the input string). Refer to the IDTNC and IDCNV monitor calls descriptions for these functions.

ACCEPTS IN	AC1:	source designator
	AC2:	format option flags (see below), 0 is the normal case
RETURNS	+1:	failure, error code in AC2, updated string pointer in AC1, if pertinent
	+2:	success, updated string pointer, if pertinent, in AC1, and the internal format date and time in AC2

The format option flags in AC2 specify the interpretation to be used when a date or time specification is ambiguous.

IDTIM Option Flags

B1(IT%NNM)	do not allow the month to be numeric and ignore B2-B3.
B2(IT%SNM)	interpret the second number in the date as the month (e.g., 6/2/76 is interpreted as Feb. 6, 1976). If this bit is off, the first number is interpreted as the month (e.g., 2/6/76 is interpreted as Feb. 6, 1976).
B3(IT%ERR)	return an error if the order of the day and month does not agree with the setting of B2(IT%SNM) even though the date can be successfully interpreted. If this bit is off, a date which can be interpreted by assuming the day and month are in the opposite order than that specified by the setting of B2(IT%SNM) will be considered valid. For example, if B2-B3 are off, 30/5/76 will be considered as a valid date.
B7(IT%NIS)	seconds cannot be included in a time specification.

B8(IT%AIS) seconds must be included in a time specification and must be preceded by a colon.

If B7-B8 are both off, seconds are optional in a time specification. If specified, seconds must be preceded by a colon.

B9(IT%NAC) colon cannot be used to separate hours and minutes.

B10(IT%AAC) colon must be used to separate hours and minutes.

If B9-B10 are both off, a colon is optional between hours and minutes.

B11(IT%AMS) when B7-B10 are off, always interpret a time specification containing one colon as hhmm:ss.

B12(IT%AHM) when B7-B10 are off, always interpret a time specification containing one colon as hh:mm and return an error if the first field is too large. This differs from B7(IT%NIS) in that seconds can be included if preceded by a second colon.

If B7-B12 are all off, a time specification containing one colon is interpreted as hh:mm if the first field is small enough. Otherwise it is interpreted as hhmm:ss.

B14(IT%N24) do not allow the time to be specified in 24-hour format (e.g., 1520 for 3:20 in the afternoon) and make AM or PM specification mandatory.

B15(IT%NTM) do not allow the time specification to include AM,PM,NOON, or MIDNIGHT.

B16(IT%NTZ) do not allow a time zone to be specified.

If AC2 is 0, the IDTIM call accepts any reasonable date and time formats. The following are examples of valid dates:

```
6-FEB-76
FEB-6-76
FEB 6 76
FEB 6, 1976
6 FEB 76
6/2/1976
2/6/76
```

Below are examples of valid times:

```
1:12:13
1234
16:30             (4:30PM)
1630
1234:56
1:56AM
1:56-EST
1200NOON
12:00:00AM        (midnight)
11:59:59AM-EST    (late morning)
12:00:01AM        (early morning)
```

"AM" or "PM" can follow a time specification that is not greater than 12:59:59. "NOON" or "MIDNIGHT" can follow 12:00:00. Any time specification can be followed by a dash and a time zone. The time zones are: EST and EDT (Eastern), CST and CDT (Central), MST and MDT (Mountain), PST and PDT (Pacific), AST and ADT (Atlantic), YST and YDT (Yukon), HST and HDT (Hawaii), BST and BDT (Bering), GMT, GST, and GDT (Greenwich), DAYLIGHT (local daylight savings), STD or STANDARD (local standard).

All strings (e.g., months, time zones, AM-PM-NOON-MIDNIGHT) can be represented by any non-ambiguous abbreviation (e.g., D=DECEMBER, M=MIDNIGHT).

Spaces are ignored before and between fields whenever they do not terminate the input string. This means spaces are not allowed before colons, AM,PM,NOON, and MIDNIGHT, the dash before the time zone, or the time zone. A tab is also allowed between the date and time.

The input string can be terminated by any non-alphanumeric character.

IDTIM ERROR MNEMONICS:

DILFX1: invalid date format

TILFX1: invalid time format

DATEX1: year out of range

DATEX3: day of month too large

DATEX5: date out of range

All I/O errors are also possible. These errors cause software interrupts or process terminations as described under the BIN call.

IDTNC JSYS 231

Inputs the date and/or the time and converts it into separate numbers for the local year, month, day, or time. The IDTNC call allows the date or time to be entered separately, which is not possible with the IDTIM JSYS because neither one can be converted to the internal format without converting the other. (Refer to Section 2.8.2.)

ACCEPTS IN AC1: source designator

AC2: format option flags
In addition to the flags described in the IDTIM call, the flags below can also be specified:

B0(IT%NDA) do not input the date and ignore B1-B3. If IT%NDA is off, the date must be input.

B6(IT%NTI) do not input the time and ignore B7-B16. If IT%NTI is off, the time must be input.

RETURNS +1: failure, error code in AC2, updated string pointer, if pertinent, in AC1

+2: success, updated string pointer, if pertinent, in AC1

If the date was input,
AC2 contains the year in the left half, and the month (0=January) in the right half.
AC3 contains the day of the month (0=first day) in the left half, and the day of the week (0=Monday) in the right half.

If the time was input,
AC4 contains

B0(IC%DAS)	on if a time zone was input (for compatibility with the ODCNV call).
B1(IC%ADS)	on if a daylight savings time zone was input.
B2(IC%UTZ)	on if a time zone was input.
B3(IC%JUD)	on if a number in Julian day format was input.
B12-B17 (IC%TMZ)	the time zone if one was input, or the local time zone if none was input. (Refer to Section 2.8.2 for the time zones.)
B18-B35 (IC%TIM)	time as seconds since midnight.

IDTNC ERROR MNEMONICS:

DILFX1: invalid date format

TILFX1: invalid time format

All I/O errors are also possible. These errors cause software interrupts or process terminations as described under the BIN call description.

The IDTNC call does not detect certain errors in date input, such as day 31 of a 30-day month. These errors are detected by the IDCNV call.

IIC JSYS 132

Initiates software interrupts on the specified channels in a process. (Refer to Section 2.5.)

ACCEPTS IN AC1: process handle

AC2: 36-bit word
Bit n on means initiate a software interrupt on channel n.

RETURNS +1: always

Generates an illegal instruction interrupt on error conditions below.

IIC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

INLNM JSYS 503

Returns a logical name that is defined either for this job or for the system. (Refer to Section 2.2.2 and CRLNM and LNMST monitor calls.)

ACCEPTS IN AC1: function code in the left half, and index into the table of defined logical names in the right half

AC2: pointer to the string for storing the logical name

RETURNS +1: failure, error code in AC1

+2: success, updated string pointer in AC2

The available functions are:

Code	Symbol	Meaning
0	.INLJB	List the logical names defined for this job
1	.INLSY	List the logical names defined for the system

INLNM ERROR MNEMONICS:

INLNX1: index is beyond end of logical name table

INLNX2: invalid function

JFNS JSYS 30

Returns the file specification currently associated with the JFN.

ACCEPTS IN AC1: destination designator where the ASCIZ string is to be written

AC2: indexable file handle (refer to GTJFN), or pointer to string

AC3: format control bits to be used when returning the string, or 0

AC4: pointer to string containing prefix of file specification attribute

RETURNS +1: always, updated string pointer, if pertinent, in AC1

AC2 can have one of two formats, depending on B26(JS%PTR) in AC3. The first format is a word with either 0 or the flag bits returned from GTJFN in the left half and the JFN in the right half. When the left half is 0, the string returned is the exact specification associated with the JFN. If the given JFN is associated only with a file specification (i.e., it was obtained with B12(GH%OFG) on in the GTJFN call), the string returned contains null fields for nonexistent fields or fields containing wildcards, and actual values for existent fields. When the left half is nonzero, the string returned contains wildcard characters for appropriate fields and 0, -1, or -2 as a generation number if the corresponding bit is on in the call.

The second format (allowed only if B26(JS%PTR) of AC3 is on) is a pointer to the string to be returned. This string is one field of a file specification. The field is determined by the first nonzero 3-bit field in AC3 or by the setting of B27(JS%ATR) or B28(JS%AT1) in AC3. For example, if bits 6-8 (JS%NAM) of AC3 are nonzero, then the string is interpreted as a filename field. If B27(JS%ATR) is on, the string is interpreted as a file specification attribute. If B28(JS%AT1) is on, the string is concatenated to the string pointed to by AC4, and a colon is inserted between the two strings. In all cases, the string is output to the destination designator, and the appropriate punctuation is added.

AC3 contains control bits for formatting the string being returned. B0-B20 are divided into 3-bit bytes, each byte representing a field in the file specification. The value of the byte indicates the output for that field. The values are:

0	(.JSNOF)	do not output this field
1	(.JSAOF)	always output this field
2	(.JSSSD)	suppress this field if it is the system default

The bits that can be set in AC3 are as follows:

B0-B2(JS%DEV)	output for device field
B3-B5(JS%DIR)	output for directory field
B6-B8(JS%NAM)	output for filename field (2 is illegal)
B9-B11(JS%TYP)	output for file type field (2 is illegal)
B12-B14(JS%GEN)	output for generation number field
B15-B17(JS%PRO)	output for protection field
B18-B20(JS%ACT)	output for account field
B21(JS%TMP)	return ;T if appropriate
B22(JS%SIZ)	return size of file in pages
B23(JS%CDR)	return creation date
B24(JS%LWR)	return date of last write
B25(JS%LRD)	return date of last read
B26(JS%PTR)	AC2 contains pointer to the string being returned
B27(JS%ATR)	return file specification attributes if appropriate
B28(JS%AT1)	return the specific specification attribute whose prefix is indicated by the string pointed to in AC4. This bit is used when a program is processing attributes one at a time. If JS%ATR is also set, all attributes will be returned.
B32(JS%PSD)	punctuate the size and date fields
B33(JS%TBR)	tab before all fields returned, except for first field
B34(JS%TBP)	tab before all fields that may be returned (i.e., fields whose value is given as 1 or 2), except for first field
B35(JS%PAF)	punctuate all fields from device through ;T

If B32-B35 are 0, punctuation between fields is not used.

If AC3 is 0, the string is output in the format

```
dev:<directory>name.typ.gen;T
```

with fields the same as system defaults not returned and the protection (;P) and account (;A) attributes returned if B9(GJ%PRO) and B10(GJ%ACT) in AC2 are on. The temporary attribute (;T) is not returned if the JFN is not associated with a file (refer to GJ%OFG in the GTJFN description) or the file is not temporary.

The punctuation used on each field is shown below. (The punctuation is underscored.)

```
dev:<directory>name.typ.gen;attribute
,size,creation date,write date,read date
```

The GTJFN or GNJFN monitor call is used to associate a JFN with a given file specification string.

Generates an illegal instruction interrupt on error conditions below.

JFNS ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
IOX11:	quota exceeded or disk full

KFORK JSYS 153

Kills one or more processes. When a process is killed, all private memory acquired by the process and its Process Storage Block are released. Also, any JFNs the process has created are released, and any terminal interrupt assignments that were acquired from another process are passed back. (Note that because the process is deleted asynchronously, a page of a file mapped into a lower process may not be unmapped before the KFORK call returns.)

ACCEPTS IN AC1: process handle

RETURNS +1: always, unless the current process attempts to kill itself

The KFORK call will not release a process handle that identifies a process already killed by another process. In this case, the RFRKH call must be used to release the handle.

The CFORK monitor call can be used to create an inferior process.

Generates an illegal instruction interrupt on error conditions below.

KFORK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

KFRKX1: illegal to kill top level process

KFRKX2: illegal to kill self

LGOUT JSYS 3

Kills the specified job and appends an accounting entry to the accounting data file. However, no entry is appended if the job was never logged in (i.e., a CTRL/C was typed, but no login occurred).

ACCEPTS IN AC1: number of the job to be logged out, or -1 for the current job

RETURNS +1: failure, error code in AC1

+2: success

When a specific job number is given in AC1, it must refer to either a PTY job controlled by the current job or a job logged in under the same user name as the current job. Otherwise, to give a specific job number, the process must have WHEEL or OPERATOR capability enabled. An argument of -1 must be given if the current job wishes to kill itself (i.e., the job number given cannot be the same as the current job). Note that this monitor call does not return if the argument in AC1 is -1.

The LGOUT monitor call outputs the time used (both CPU and console), the job number, and the current date and time. This information is output on the terminal to which the job being logged out is attached.

LGOUT ERROR MNEMONICS:

LOUTX1: illegal to specify job number when logging out own job

LOUTX2: invalid job number

LOUTX3: WHEEL or OPERATOR capability required

LOUTX4: LOG capability required

LOUTX5: illegal to log out job 0

LNMST JSYS 504

Translates a logical name to its original definition string. (Refer to Section 2.2.2 and the CRLNM and INLNM monitor calls descriptions.)

ACCEPTS IN	AC1:	function code
	AC2:	pointer to the logical name
	AC3:	pointer to the string where the original logical name definition is to be written
RETURNS	+1:	failure, error code in AC1
	+2:	success, updated string pointer in AC3

The codes for the functions are as follows:

0	.LNSJB	Obtain the job-wide definition of the logical name.
1	.LNSSY	Obtain the system definition of the logical name.

LNMST ERROR MNEMONICS:

GJFX22: insufficient system resources (Job Storage Block full)

LNSTX1: no such logical name

LNSTX2: invalid function

LOGIN JSYS 1

Logs a job into the system. Useful for logging in from an idle terminal on which a CTRL/C has been typed.

ACCEPTS IN	AC1:	36-bit user number under which user will log in
	AC2:	pointer to beginning of password string
	AC3:	account number in bits 3-35 if bits 0-2 are 5. Otherwise contains a pointer to an account string. If a null byte is not seen, the string is terminated after 39 characters are processed.
RETURNS	+1:	failure, error code in AC1
	+2:	success, date and time of last login (in internal system format; refer to Section 2.8.2) in AC1, and updated string pointers, if pertinent, in AC2 and AC3.

The LOGIN monitor call does not require a password if the controlling terminal is a pseudo-terminal and the controlling job either has the WHEEL or OPERATOR capability enabled or is logged in as the same user being logged in for this job.

If the call is successful, an accounting entry is appended to the accounting data file. If the account validation facility is enabled, the LOGIN call verifies either the account given or the default account of the user being logged in.

LOGIN ERROR MNEMONICS:

LGINX1: invalid account identifier

LGINX2: directory is "files-only" and cannot be logged in to

LGINX3: internal format of directory is incorrect

LGINX4: invalid password

LGINX5: job is already logged in

LGINX6: no more job slots available for logging in

LPINI JSYS 547

Loads the direct access Vertical Formatting Unit (VFU) or translation Random Access Memory (RAM) for the line printer. This call is executed at system startup by the program that configures the system.

ACCEPTS IN AC1: JFN of file containing VFU or RAM

AC2: status bits in the left half, and function code in the right half

AC3: unit number of line printer

RETURNS +1: always

The following status bit is currently defined.

B0(MO%LCP) Line printer is a lowercase printer.

The available functions are as follows:

Code	Symbol	Meaning
32	.MOLVF	Load the VFU from the file indicated by the given JFN.
34	.MOLTR	Load the translation RAM from the file indicated by the given JFN.

The LPINI monitor call requires the process to have WHEEL or OPERATOR capability enabled. In addition, the line printer must not be opened by any process when this call is executed. If a condition occurs that prevents the VFU or RAM from being loaded (e.g., the line printer is off line), the name of the file will be stored. The VFU or RAM will then be loaded automatically the next time a process performs output to the line printer.

Generates an illegal instruction interrupt on error conditions below.

LPINI ERROR MNEMONICS:

LPINX1: invalid unit number

LPINX2: WHEEL or OPERATOR capability required

LPINX3: illegal to load RAM or VFU while device is OPEN

MRECV JSYS 511

Retrieves an IPCF (Inter-Process Communication Facility) message from the process' input queue. Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

ACCEPTS IN AC1: length of packet descriptor block

AC2: address of packet descriptor block

RETURNS +1: failure, error code in AC1

+2: success. The packet is retrieved and placed into the block indicated by word .IPCFP of the packet descriptor block. AC1 contains the length of the next entry in the queue in the left half and the flags from the next packet in the right half. This returned word is called the associated variable of the next entry in the queue. If the queue is empty, AC1 contains 0.

The format of the packet descriptor block is as follows:

Word	Symbol	Meaning
0	.IPCFL	Flags. (Refer to the MSEND call description.)
1	.IPCFS	PID of sender. The caller does not supply this PID; the system fills it in when the packet is retrieved.
2	.IPCFR	PID of receiver. This PID can be one of three values: a specific PID, -1 to retrieve messages for any PID belonging to this process, or -2 to retrieve messages for any PID belonging to this job. When -1 or -2 is supplied, messages are not retrieved in any particular order except that messages from a specific PID are returned in the order in which they were received.
3	.IPCFP	Pointer to block where message is to be placed (length of message in the left half and address where message is to be placed in the right half).

4	.IPCFD	User number of sender.
5	.IPCFC	Enabled capabilities of sender.
6	.IPCSD	Directory number of sender's connected directory.
7	.IPCAS	Account string of sender. The caller supplies a pointer to the block where the account is to be placed.

The caller (i.e., receiver) does not supply the information in words 4 through 7; the system fills in the words when the packet is retrieved. These words describe the sender at the time the message was sent and permit the receiver to validate messages.

Refer to the MSEND call description for the flags that can be set in word .IPCFL of the packet descriptor block.

MRECV ERROR MNEMONICS:

IPCFX1: length of packet descriptor block cannot be less than 4

IPCFX2: no message for this PID

IPCFX3: data too long for user's buffer

IPCFX4: receiver's PID invalid

IPCFX5: receiver's PID disabled

IPCFX7: receiver quota exceeded

IPCFX8: IPCF free space exhausted

IPCF10: WHEEL capability required

IPCF11: WHEEL or IPCF capability required

IPCF12: no free PID's available

IPCF13: PID quota exceeded

IPCF14: no PID's available to this job

IPCF15: no PID's available to this process

IPCF16: receive and message data modes do not match

IPCF19: no PID for [SYSTEM]INFO

IPCF24: invalid message size

IPCF25: PID does not belong to this job

IPCF26: PID does not belong to this process

IPCF27: PID is not defined

IPCF28: PID not accessible by this process

IPCF29: PID already being used by another process

IPCF31: invalid page number

IPCF32: page is not private

IPCF34: cannot receive into an existing page

MSEND JSYS 510

Sends an IPCF (Inter-Process Communication Facility) message. The message is in the form of a packet and can be sent to either the specified PID or the system process <SYSTEM>INFO. Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

ACCEPTS IN AC1: length of packet descriptor block

AC2: address of packet descriptor block

RETURNS +1: failure, error code in AC1

+2: success. The packet is sent to the receiver's input queue. Word .IPCFS of the packet descriptor block is updated with the sender's PID. This updating is done in case the PID was being defaulted or created by this call.

The format of the packet descriptor block is as follows:

Word	Symbol	Meaning
0	.IPCFL	Flags. (See below.)
1	.IPCFS	PID of sender, or 0 if no PID exists for sender. This word will be 0 if the caller is creating a PID (i.e., flag bit IP%CPD is on).
2	.IPCFR	PID of receiver, or 0 if receiver is <SYSTEM>INFO.
3	.IPCFP	Pointer to message block (length of message in the left half and starting address of message in the right half). When a packet is sent to <SYSTEM>INFO, the message block contains the request being made. (See below.)

The following flags are defined in word .IPCFL of the packet descriptor block. These flags can be set on both the MSEND and MREVC calls.

Flags Set By Caller

B0(IP%CFB) Do not block process if there are no messages in the queue. If this bit is set, an error is given if there are no messages.

B1(IP%CFS) Use, as the sender's PID, the PID obtained from the address specified in word .IPCFS.

B2(IP%CFR)	Use, as the receiver's PID, the PID obtained from the address specified in word .IPCFR.
B3(IP%CFO)	Allow one send request above the quota. (The default send quota is 2.)
B4(IP%TTL)	Truncate the message, if it is larger than the space reserved. If this bit is not set, an error is given if the message is too large.
B5(IP%CPD)	Create a PID to use as the sender's PID and return it in word .IPCFS of the packet descriptor block.
B6(IP%JWP)	Make the created PID be job wide (i.e., permanent until the job logs out). If this bit is not set, the PID is temporary until the process executes the RESET monitor call. If B5(IP%CPD) is not set, B6 is ignored.
B7(IP%NOA)	Do not allow other processes to use the created PID. If B5(IP%CPD) is not set, B7 is ignored.
B18(IP%CFP)	The packet is privileged. (This bit can be set only by a process with WHEEL capability enabled.) When a privileged sender sets this bit, the MRECV and MUTIL calls return it set for any reply. An error is given if this bit is set by the sender and the receiver is not privileged.
B19(IP%CFV)	The packet is a page of data. Word .IPCFP of the packet descriptor block contains 1000 in the left half and the page number in the right half. The page the packet is being sent to, or is being received into, must be private.

Flags Returned After Call

B20(IP%CFZ)	A zero-length message was sent, and the packet consists of only the packet descriptor block.
B24-B29 (IP%CFE)	Error code field for errors encountered by <SYSTEM>INFO during a send or receive request.

Code	Symbol	Meaning
15	.IPCPI	insufficient privileges
16	.IPCUF	invalid function
67	.IPCSN	<SYSTEM>INFO needs name
72	.IPCFF	<SYSTEM>INFO free space exhausted
74	.IPCBP	PID has no name or is invalid
75	.IPCDN	duplicate name has been specified
76	.IPCNN	unknown name has been specified
77	.IPCEN	invalid name has been specified

B30-B32 (IP%CFC)	System and sender code. This code can be set only by a process with WHEEL capability enabled. The system returns the code so that a nonprivileged user can examine it.

Code	Symbol	Meaning
1	.IPCCC	sent by <SYSTEM>IPCF
2	.IPCCF	sent by system-wide <SYSTEM>INFO
3	.IPCCP	sent by receiver's <SYSTEM>INFO

B33-B35 (IP%CFM) — Field for return of special messages. This field can be set only by a process with WHEEL capability enabled. The system returns the information so that a nonprivileged user can examine it.

Code	Symbol	Meaning
1	.IPCFN	Process' input queue contains a packet that could not be delivered to intended PID.

When the MSEND call is used to send a packet to <SYSTEM>INFO, the message portion of the packet (i.e., the first three words) contains the request. This request has the following format:

Word	Symbol	Meaning
0	.IPCI0	user-defined code in the left half and the function (see below) <SYSTEM>INFO is to perform in the right half. The user-defined code is used to associate the response from <SYSTEM>INFO with the appropriate request.
1	.IPCI1	PID that is to receive a duplicate of the response from <SYSTEM>INFO. If this word is 0, the response is sent only to the originator of the request.
2	.IPCI2	argument for the requested function. (See below.)

The functions that can be requested of <SYSTEM>INFO, along with their arguments, are as follows:

Function	Argument	Meaning
.IPCIW	name	Return the PID associated with the specified name. The PID is returned in word .IPCI1.
.IPCIG	PID	Return the name associated with the specified PID. The name is returned in word .IPCI1.
.IPCII	name in ASCIZ	Assign the specified name to the PID belonging to the process making the request. The temporary or permanent status of the PID is specified by flag bit IP%JWP(B6) when the PID was originally created.
.IPCIJ	name in ASCIZ	Identical to the .IPCII function.
.IPCIS		Disassociates all PIDs with names. Used by the monitor on a RESET or LGOUT monitor call. This function is not available to user programs.

MSEND ERROR MNEMONICS:

IPCFX1:	length of packet descriptor block cannot be less than 4
IPCFX3:	data too long for user's buffer
IPCFX4:	receiver's PID invalid
IPCFX5:	receiver's PID disabled
IPCFX6:	send quota exceeded
IPCFX7:	receiver quota exceeded
IPCFX8:	IPCF free space exhausted
IPCFX9:	sender's PID invalid
IPCF10:	WHEEL capability required
IPCF11:	WHEEL or IPCF capability required
IPCF12:	no free PID's available
IPCF13:	PID quota exceeded
IPCF14:	no PID's available to this job
IPCF15:	no PID's available to this process
IPCF16:	receive and message data modes do not match
IPCF19:	no PID for [SYSTEM]INFO
IPCF24:	invalid message size
IPCF25:	PID does not belong to this job
IPCF26:	PID does not belong to this process
IPCF27:	PID is not defined
IPCF28:	PID not accessible by this process
IPCF29:	PID already being used by another process
IPCF31:	invalid page number
IPCF32:	page is not private

MSFRK JSYS 312

Starts a process in monitor mode. The MSFRK call is legal only if it is called from monitor mode or if the process has WHEEL or OPERATOR capability enabled. This call allows job 0 to create multiple processes for handling various asynchronous monitor tasks.

ACCEPTS IN AC1: process handle

AC2: 36-bit PC word, with user mode and other flags in the left half and the virtual address in the right half

RETURNS +1: always

Because the starting context of the process is undefined, the process being started should execute the following sequence of instructions at its starting address:

```
FBGN:   MOVSI 1,UMODF       ;fake user PC
        MOVEM 1,FPC         ;simulate the JSYS call
        MCENTR              ;establish usual top-level JSYS context
```

Generates an illegal instruction interrupt on error conditions below.

MSFRK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

CAPX1: WHEEL or OPERATOR capability required

MSTR JSYS 555

Performs various structure-dependent functions. These functions include mounting and dismounting structures, incrementing and decrementing mount counts for structures, and setting and obtaining the status of structures. Some functions of the MSTR monitor call require the process to have WHEEL or OPERATOR capability enabled. A process with MAINTENANCE capability enabled can obtain the status of a disk unit (functions .MSRNU and .MSRUS).

ACCEPTS IN AC1: length of the argument block in the left half and function code in the right half

AC2: address of the argument block

RETURNS +1: always, with some functions returning data in the argument block. (Refer to individual function descriptions below.)

Generates an illegal instruction interrupt on all error conditions.

The available functions are summarized below.

Function Code	Symbol	Privileged	Meaning
0	.MSRNU	Yes @	Return the status of the next disk unit.
1	.MSRUS	Yes	Return the status of the given disk unit.
2	.MSMNT	Yes	Mount the given structure.
3	.MSDIS	Yes	Dismount the given structure.
4	.MSGSS	No	Return the status of the given structure.
5	.MSSSS	Yes	Change the status of the given structure.
6	.MSINI	Yes	Initialize the given structure.
7	.MSIMC	No	Increment the mount count for the given structure.
10	.MSDMC	No	Decrement the mount count for the given structure.
11	.MSGSU	No	Return the job numbers of the users of the given structure.

Obtaining the Status of the Next Disk Unit - .MSRNU

This function returns the status of the next disk unit on the system. The next disk unit is determined by searching the current channel and looking for the next physical unit on that channel.

The .MSRNU function accepts the channel, controller, and unit numbers in the first three words of the argument block. The first time this function is executed, the value for each of these numbers is -1. After successful completion of this function, the channel, controller, and unit numbers are updated, and the software information about the disk drive is returned in the argument block. To locate all drives available for mounting structures, the channel, controller, and unit numbers returned from one .MSRNU function call are supplied on the next one until all units on all channels have been searched. When all units have been searched, the MSTR monitor call returns error MSTX18.

The format of the argument block, whose length is .MSRLN, is as follows:

Word	Symbol	Meaning
0	.MSRCH	Channel number (0-7)
1	.MSRCT	Controller number (reserved for future use, must be -1)
2	.MSRUN	Unit number (0-7)

3	.MSRST	Returned software status of unit. The following status bits are defined: B0(MS%MNT) Unit is part of a mounted structure B2(MS%DIA) Unit is being used by an on-line diagnostic program B3(MS%OFL) Unit is off line B4(MS%ERR) Unit has an error that was detected during reading B5(MS%BBB) Unit has a bad BAT block. If this bit is on, the data returned in word .MSRSN (word 4) and in words .MSRNS through .MSRFI (words 6 through 20) is indeterminate. B6(MS%HBB) Unit has a bad HOME block B7(MS%WLK) Unit is write locked B9-B17 (MS%TYP) Type of disk unit 1 .MSRP4 RP04 5 .MSRP5 RP05 6 .MSRP6 RP06
4	.MSRSN	Pointer to ASCIZ string in which to store the structure name. This pointer is updated on return.
5	.MSRSA	Pointer to ASCIZ string in which to store the structure alias. The alias is usually the same as the structure name. The alias is returned, and the pointer updated, only if the structure is on line.
6	.MSRNS	Logical unit number within the structure of this unit in the left half, and number of units in the structure in the right half.
7	.MSRSW	Number of pages for swapping on this structure.
10-12	.MSRUI	Unit ID (3 words of ASCII)
13-15	.MSROI	Owner ID (3 words of ASCII)
16-20	.MSRFI	File system ID (3 words of ASCII)
21	.MSRSP	Number of sectors per page
22	.MSRSC	Number of sectors per cylinder
23	.MSRPC	Number of pages per cylinder
24	.MSRCU	Number of cylinders per unit
25	.MSRSU	Number of sectors per unit
26	.MSRBT	Number of bit words in bit table per cylinder

The following errors are possible on the failure of this function.

MSTRX2:	WHEEL or OPERATOR capability required
MSTRX3:	argument block too small
MSTX14:	invalid channel number
MSTX15:	invalid unit number
MSTX16:	invalid controller number
MSTX18:	no more units in system
MSTX27:	specified unit is not a disk
CAPX2:	WHEEL, OPERATOR, or MAINTENANCE capability required

Obtaining the Status of a Given Disk Unit - .MSRUS

This function returns the status of the given disk unit. It accepts the channel, controller, and unit numbers in the first three words of the argument block. After successful completion of this function, the channel, controller, and unit numbers are unchanged, and the software information about the given disk unit is returned in the argument block.

The difference between this function and the .MSRNU function is that .MSRUS does not search for the next disk unit but rather returns the status for the given unit. The .MSRNU function searches for the next disk unit and returns the status for that unit.

The format of the argument block and the errors possible on the failure of this function are the same as described for the .MSRNU function.

Mounting a Given Structure - .MSMNT

This function brings the given structure on line and normally makes it available for general use. Any structure other than the public structure PS: must be brought on line with this function. (The public structure PS: is brought on line during phe system startup procedure.)

It is recommended that the .MSRNU (Read Next Unit) function be given first to locate all units in the structure. Then the .MSMNT (Mount Structure) function can be given to read and verify the HOME blocks of each unit and to mount the structure. If one or more units of the structure are write-locked, the structure cannot be mounted and an error is given.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MSTNM	Pointer to the ASCIZ string containing the name of the structure.
1	.MSTAL	Pointer to the ASCIZ string containing the alias of the structure.

2	.MSTFL	Flag bits in the left half, and the number of units in the structure (.MSTNU) in the right half. The bits that can be set in the left half are:

B0(MS%NFH) If one of the HOME blocks is incorrect, do not fix it and do return an error. If this bit is off and one of the HOME blocks is incorrect, the correct block is copied into the bad HOME block and the mounting procedure continues.

B1(MS%NFB) If one of the BAT (Bad Allocation Table) blocks is incorrect, do not fix it and do return an error. If this bit is off and one of the BAT blocks is incorrect, the correct block is copied into the bad BAT block and the mounting procedure continues.

B2(MS%XCL) Mount the structure for exclusive use by this job. This bit is set by a system program when it initializes or reconstructs a structure. If this bit if off, the structure is mounted for general use.

B3(MS%IGN) Ignore correctable errors in the bit table and in the root directory on this structure. This bit is set by a system program when it reconstructs the root directory on a structure or rebuilds the bit table. If this bit is off and an error is detected, this function returns an error.

3	.MSTUI	Beginning of unit information for each unit in the structure. The information is 3 words long per unit, and the symbol for this length is .MSTNO. The first 3-word block is for logical unit 0, and the last 3-word block is for the last logical unit (.MSTNU-1). The offsets into the 3-word block are:

0	.MSTCH	Channel number of unit
1	.MSTCT	Controller number of unit (currently must be -1)
2	.MSTUN	Unit number of unit

After successful completion of this function, the given structure is mounted and available for general use (unless bit MS%XCL was on in word .MSTFL of the argument block).

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required

MSTRX3: argument block too small

MSTRX4: insufficient system resources

MSTRX5: drive is not on line

MSTRX6: home blocks are bad

MSTRX7: invalid structure name

MSTRX8: could not get OFN for ROOT-DIRECTORY

MSTRX9: could not MAP ROOT-DIRECTORY

MSTX10: ROOT-DIRECTORY bad

MSTX11: could not initialize Index Table

MSTX12: could not OPEN Bit Table File

MSTX13: backup copy of ROOT-DIRECTORY is bad

MSTX14: invalid channel number

MSTX15: invalid unit number

MSTX16: invalid controller number

MSTX17: all units in a structure must be of the same type

MSTX19: unit is already part of a mounted structure

MSTX20: data error reading HOME blocks

MSTX23: could not write HOME blocks

MSTX25: invalid number of swapping pages

MSTX27: specified unit is not a disk

MSTX30: incorrect Bit Table counts on structure

MSTX34: unit is write-locked

MONX01: insufficient system resources

Dismounting a Given Structure - .MSDIS

This function indicates that the given structure can be removed from the system. Any mounted structure other than the public structure PS: can be dismounted with this function. (The public structure PS: is dismounted at system shutdown.)

Files that are open at the time this function is executed become inaccessible, and the jobs that had the files open receive an error if they reference them. Jobs that have mounted the structure or have connected to or accessed a directory on the structure receive an informational message on the terminal. This message is

[STRUCTURE name: HAS BEEN DISMOUNTED]

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MSDNM	Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure.

After successful completion of this function, the given structure is dismounted and can be physically removed from the system.

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required

MSTRX3: argument block too small

MSTX21: structure is not mounted

MSTX24: illegal to dismount the Public Structure

Obtaining the Status of a Given Structure - .MSGSS

This function returns the status of a mounted structure. The caller supplies the designators for the structure and for the storage of the structure's physical ID. After successful completion of the call, data is returned in the appropriate words in the argument block.

The format of the argument block, whose length is .MSGLN, is as follows:

Word	Symbol	Meaning	
0	.MSGSN	Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure.	
1	.MSGST	Returned status word. The status bits are:	
		B0(MS%PS)	This structure is a public structure.
		B1(MS%DIS)	This structure is being dismounted and no further mount count increments are allowed.
		B2(MS%DOM)	This structure is a domestic structure.
		B3(MS%PPS)	This structure is the primary public structure.
		B4(MS%INI)	This structure is being initialized.

2	.MSGNU	Number of units in structure.
3	.MSGMC	Mount count for this structure. This value is determined by the number of .MSIMC (Increment Mount Count) functions given for this structure by all users since the structure was mounted.
4	.MSGFC	Open file count (i.e., number of open files) for this structure.
5	.MSGSI	Pointer to ASCIZ string in which to store the structure's physical ID.

After successful completion of this function, the status of the given structure is returned in the appropriate words of the argument block, and the pointer to the physical ID is updated to reflect the returned string.

The following errors are possible on the failure of this function.

MSTRX3: argument block too small

MSTX21: structure is not mounted

Changing the Status of a Given Structure - .MSSSS

This function changes the status of a mounted structure. The caller can change only two of the status bits in the structure's status word: status of being dismounted and status of being domestic.

The format of the argument block, whose length is .MSSLN is:

Word	Symbol	Meaning
0	.MSSSN	Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure.
1	.MSSST	Word containing the new values for the bits being changed.
2	.MSSMW	Mask containing the bits being changed. The bits that can be changed are B1(MS%DIS) Structure is being dismounted. B2(MS%DOM) Structure is domestic.

After successful completion of this function, the status of the given structure is changed according to the data supplied in the argument block.

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required

MSTRX3: argument block too small

MSTX21: structure is not mounted

MSTX22: illegal to change specified bits

Initializing a Given Structure - .MSINI

This function creates a new structure or repairs an existing structure during normal system operation. The caller has the option of creating a new file system, reconstructing the root directory, writing a new set of HOME blocks on the structure, or rebuilding the index block.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MSINM	Pointer to ASCIZ string containing the name of the structure.
1	.MSIAL	Pointer to ASCIZ string containing the alias of the structure.
2	.MSIFL	Flag bits in B0-B11, function value (MS%FCN) in B12-B17, and number of units in structure (.MSINU) in B18-B35. The flag bits are: B0(MS%NFH) Do not fix HOME block if one is incorrect and do return an error. This bit can be on only with function .MSRRD. (See below.) B1(MS%NFB) Do not fix BAT block if one is incorrect and do return an error. B2(MS%XCL) Mount this structure for exclusive use by this job. If this bit is off, the structure is mounted for general use. B3(MS%IGN) Ignore errors in the bit table and in the root directory on this structure. If this bit is on, B2(MS%XCL) must also be on. The function values that can be given are: 1 .MSCRE Create a new file system 2 .MSRRD Reconstruct the root directory 3 .MSWHB Write a new set of HOME blocks 4 .MSRIX Rebuild the index table
3-5	.MSISU	Beginning of unit information for each unit in the structure. The information is 3 words long per unit, and the symbol for this length is .MSINO. The first 3-word block is for logical unit 0, and the last 3-word block is for the last logical unit (.MSINU-1). The offsets into the 3-word block are: 0 .MSICH Channel number of unit 1 .MSICT Controller number of unit (currently must be -1) 2 .MSIUN Unit number of unit

6	.MSIST	Status word (reserved for future use).
7	.MSISW	Number of pages for swapping on this structure.
10	.MSIFE	Number of pages for the front-end file system.
11-13	.MSIUI	Unit ID (3 words of ASCII)
14-16	.MSIOI	Owner ID (3 words of ASCII)
17-21	.MSIFI	File system ID (3 words of ASCII) (reserved for future use)
22	.MSIFB	Number of pages for the file BOOTSTRAP.BIN.

Words 6 through 16 (.MSIST through .MSIOI) of the argument block must be supplied when the MSTR call is being executed to create a new file system or to write a new set of HOME blocks. After successful completion of the .MSCRE function, the structure is initialized and the following directories are created:

<ROOT-DIRECTORY>
<SYSTEM>
<SUBSYS>
<ACCOUNTS>
<SPOOL>
<OPERATOR>

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required

MSTRX3: argument block too small

MSTRX4: insufficient system resources

MSTRX5: drive is not on line

MSTRX6: home blocks are bad

MSTRX7: invalid structure name

MSTRX8: could not get OFN for ROOT-DIRECTORY

MSTRX9: could not MAP ROOT-DIRECTORY

MSTX10: ROOT-DIRECTORY bad

MSTX11: could not initialize Index Table

MSTX12: could not OPEN Bit Table File

MSTX13: backup copy of ROOT-DIRECTORY is bad

MSTX14: invalid channel number

MSTX15: invalid unit number

MSTX16: invalid controller number

MSTX17: all units in a structure must be of the same type

MSTX19: unit is already part of a mounted structure

MSTX20: data error reading HOME blocks

MSTX23: could not write HOME blocks

MSTX25: invalid number of swapping pages

MSTX26: invalid number of Front-End-Filesystem pages

MSTX27: specified unit is not a disk

MSTX28: could not initialize Bit Table for structure

MSTX29: could not reconstruct ROOT-DIRECTORY

MSTX30: incorrect Bit Table counts on structure

MONX01: insufficient system resources

Incrementing the Mount Count - .MSIMC

Users indicate that they are actively using a structure by incrementing the structure's mount count. A nonzero mount count informs the operator that the structure should not be dismounted. The .MSIMC function is used to increment a structure's mount count.

The job receives an error if the given structure is in the process of being dismounted (i.e., a job has given the .MSSSS function with the MS%DIS bit on).

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MSDEV	Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure.

After successful completion of this function, the mount count of the given structure has been incremented.

The following errors are possible on the failure of this function.

MSTRX3: argument block too small

MSTX21: structure is not mounted

MSTX33: structure is unavailable for mounting

ARGX18: invalid structure name

MONX01: insufficient system resources

STDVX1: no such device

STRX01: structure is not mounted

STRX02: insufficient system resources

Decrementing the Mount Count - .MSDMC

This function indicates that the given structure is no longer being used by the job executing the call. If the job executing the call has previously incremented the mount count for this structure via the .MSIMC (Increment Mount Count) function, the mount count is decremented. If the job has not incremented the mount count, the job receives an error.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MSDEV	Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure.

The resource allocator receives an IPCF packet when the mount count for a structure is decremented. The flag word (.IPCFL) of the packet descriptor block has a code of 1(.IPCCC) in the IP%CFC field (bits 30-32). This code indicates the message was sent by the monitor. The first word of the packet data block contains the structure dismount code .IPCDS. The second word contains the number of header words and the number of the job decrementing the mount count. The third word contains the device designator of the structure. Thus,

```
.IPCFL/<.IPCCC>B32

DATA/.IPCDS
DATA+1/number of header words (2),, job number
DATA+2/device designator of structure
```

After successful completion of this function, the mount count of the structure has been decremented and the IPCF message has been sent.

The following errors are possible on the failure of this function.

MSTRX3: argument block too small

MSTX21: structure is not mounted

MSTX32: structure was not mounted

ARGX18: invalid structure name

MONX01: insufficient system resources

STDVX1: no such device

STRX01: structure is not mounted

STRX02: insufficient system resources

Obtaining the Users on a Given Structure - .MSGSU

This function returns the job numbers of the users of the given structure. Users of a structure are divided into three classes: users who have incremented the mount count (SMOUNT command), users who are connected to the structure (CONNECT command), and users who have accessed the structure (ACCESS command). The caller specifies the classes of users for which information is to be returned by setting the appropriate bits in the argument block.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MSUAL	Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure.
1	.MSUFL	Flag bits in the left half and 0 in the right half. The bits that can be set are:

B0(MS%GTA)	Return users who have accessed the structure.
B1(MS%GTM)	Return users who have incremented the mount count.
B2(MS%GTC)	Return users who are connected to the structure.

After successful execution of this function, word 1 through word n+1 (where n is the number of items returned) are updated with the following information.

Word	Symbol	Meaning
1	.MSUFL	Right half contains the number of items (n) being returned. Left half is unchanged.
2	.MSUJ1	Flag bits for the job in the left half, and number of job in the right half.
.		.
.		.
.		.
n + 1		Flag bits for the job in the left half, and number of job in the right half.

The bits returned for each job are defined as:

B0(MS%GTA)	Job has accessed structure.
B1(MS%GTM)	Job has incremented the mount count for structure.
B2(MS%GTC)	Job has connected to structure.

The following errors are possible on the failure of this function.

MSTRX1: invalid function

MSTRX3: argument block too small

STRX01: structure is not mounted

STDVX1: no such device

ARGX18: invalid structure name

MONX01: insufficient system resources

MTALN JSYS 774

Associates a given serial-numbered magnetic tape drive with the specified logical unit number. This monitor call requires the process to have WHEEL or OPERATOR capability enabled. The MTALN call is a temporary call and may not be defined in future releases.

ACCEPTS IN AC1: logical unit number of magnetic tape

AC2: decimal serial number of magnetic tape drive

RETURNS +1: always

All units are searched for the specified serial number. When it is found, the drive is associated with the given logical unit number. The original unit is now associated with the logical unit number that the specified serial-numbered drive had before it was reassigned.

Generates an illegal instruction interrupt on error conditions below.

MTALN ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

DEVX1: invalid device designator

OPNX7: device already assigned to another job

MTOPR JSYS 77

Performs various device-dependent control functions. This monitor call requires that the device either be opened or be assigned to the caller if the device is an assignable device.

Because of the device dependencies of the MTOPR call, programs written with device-independent code should not use this call unless they first check for the type of device.

ACCEPTS IN AC1: JFN of the device

AC2: function code (see below)

AC3: function arguments or address of argument block (see descriptions of individual devices)

RETURNS +1: always

The functions listed for each device apply only to that device. If a function applies to more than one device, its description is repeated for each applicable device.

MTA Functions

The functions available for magnetic tapes (MTA) are described below. Some of these functions accept arguments in AC3 (refer to the individual descriptions).

Code	Symbol	Meaning
0	.MOCLE	Clear any error flags from a previous MTOPR call.
31	.MONOP	Wait for all activity to stop.
1	.MOREW	Rewind the tape. This function waits for activity to stop before winding the tape. If sequential data is being output, the last partial buffer is written before the tape is rewound. Control returns to caller when rewinding begins.
11	.MORUL	Rewind and unload the tape. This function is identical to the .MOREW function and also unloads the tape if the hardware supports tape unloading.
10	.MOEOT	Advance forward until two sequential tape marks are seen and position tape after the first tape mark.
3	.MOEOF	Write a tape mark. This function requires that the magnetic tape be opened for write access. If sequential data is being output, the last partial buffer is written before the tape mark.
6	.MOFWR	Advance over one record in the direction away from the beginning of the tape. If sequential data is being read in the forward direction and not all of the record has been read, this function advances to the start of the next record. If sequential data is being read in the reverse direction and not all of the record has been read, this function positions the tape at the end of that record.
7	.MOBKR	Space backward over one record in the direction toward the beginning of the tape. If sequential data is being read in the forward direction and not all of the record has been read, this function positions the tape back to the start of that record. If sequential data is being read in the reverse direction and not all of the record has been read, this function positions the tape to the end of the record physically preceding that record.
16	.MOFWF	Advance to the start of the next file. This function advances the tape in the direction away from the beginning of the tape until it passes over a tape mark.
17	.MOBKF	Space backward over one file. This function moves the tape in the direction toward the beginning of the tape until it passes over a tape mark or reaches the beginning of the tape, whichever occurs first.

2	.MOSDR	Set the direction of the tape motions for read operations. This function requires AC3 to contain the desired direction. If AC3 = 0, the tape motion is forwards; if AC3 = 1, the tape motion is backwards.
26	.MORDR	Return the direction that the tape is moving during read operations. On a successful return, AC3 = 0 if the direction of the tape motion is forwards, or AC3 = 1 if the direction of the tape motion is backwards.
5	.MOSRS	Set the size of the records. This function requires AC3 to contain the desired number of bytes in the records.
15	.MORRS	Return the size of the records. On a successful return, AC3 contains the number of bytes in the records.
24	.MOSDN	Set the density. The function requires AC3 to contain the desired density: 0 .SJDDN default system density 1 .SJDN2 200 BPI 2 .SJDN5 556 BPI 3 .SJDN8 800 BPI 4 .SJD16 1600 BPI
12	.MORDN	Return the current density setting. On a successful return, AC3 contains the current density.
4	.MOSDM	Set the hardware data mode to be used when transferring data to and from the tape. This function requires AC3 to contain the desired data mode: 0 .SJDDM default system data mode 1 .SJDMC dump mode (36-bit bytes) 2 .SJDM6 SIXBIT byte mode for 7-track drives 3 .SJDMA ANSI ASCII mode (7 bits in 8-bit bytes) 4 .SJDM8 industry compatible mode
14	.MORDM	Return the hardware data mode currently being used in transfers to and from the tape. On a successful return, AC3 contains the current data mode.
20	.MOSPR	Set the parity. This function requires AC3 to contain the desired parity: 0 .SJPRO odd parity 1 .SJPRE even parity
21	.MORPR	Return the current parity. On a successful return, AC3 contains the current parity.
27	.MOSID	Set the reel identification of the tape mounted. The process must have WHEEL or OPERATOR capability enabled. This function requires AC3 to contain the desired 36-bit reel ID.

13	.MOERS	Erase three inches of tape (i.e., erase gap). This function requires that the magnetic tape be opened for write access.
22	.MONRB	Return number of bytes remaining in the current record. On a successful return, AC3 contains the number of bytes remaining. This function is only meaningful during sequential I/O.
23	.MOFOU	Force any partial records to be written during sequential output.
25	.MOINF	Return information about the tape. This function requires AC3 to contain the address of the argument block in which the information is to be returned. The format of the argument block is as follows:

Word	Symbol	Meaning
0	.MOICT	Length of argument block to be returned (not including this word)
1	.MOITP	MTA type code
2	.MOIID	MTA reel ID
3	.MOISN	Channel, controller, and unit in the left half and serial number in the right half.
4	.MOIRD	Number of reads done
5	.MOIWT	Number of writes done
6	.MOIRC	Record number from beginning of tape
7	.MOIFC	Number of files on tape
10	.MOISR	Number of soft read errors
11	.MOISW	Number of soft write errors
12	.MOIHR	Number of hard read errors
13	.MOIHW	Number of hard write errors

30	.MOIEL	Inhibit error logging for the tape.

LPT Functions

The functions available for line printers (LPT) are described below. Some of these functions accept the address of an argument block in AC3. The first word of the argument block contains the length (including this word) of the block. Remaining words of the block contain arguments for the particular function.

Code	Symbol	Meaning
27	.MOPSI	Enable for a software interrupt on nonfatal device conditions. Examples of these conditions are: 1. Device changed from off line to on line. 2. Device changed from on line to off line. 3. Device's page counter has overflowed. Other device errors or software conditions are not handled by this function; instead they cause a software interrupt on channel 11 (.ICDAE).

Argument Block

E: 3
E+1: interrupt channel number
E+2: flags. The following flag is defined:

B0(MO%MSG) Suppress standard CTY device messages.

31	.MONOP	Wait for all activity to stop. This function blocks the process until all data has actually been sent to the printer and has been printed. Because this function is transferring data, it can return an IOX5 data error.
32	.MOLVF	Load the line printer's VFU (Vertical Formatting Unit) from the file indicated in the argument block. Argument Block E: 2 E+1: JFN of the file containing the VFU The system opens the file for input with a byte size of 18 bits. It closes the file and releases the JFN when the loading of the VFU is complete.
33	.MORVF	Read the name of the current VFU file stored in the monitor's data base. Argument Block E: 3 E+1: pointer to destination area for ASCIZ name string E+2: number of bytes in destination area
34	.MOLTR	Load the line printer's translation RAM (Random Access Memory) from the file indicated in the argument block. Argument Block E: 2 E+1: JFN of the file containing the translation RAM The system opens the file for input with a byte size of 18 bits. It closes the file and releases the JFN when the loading of the translation RAM is complete.
35	.MORTR	Read the name of the current translation RAM file stored in the monitor's data base. Argument Block E: 3 E+1: pointer to destination area for ASCIZ name string E+2: number of bytes in destination area

36 .MOSTS Set the status of the line printer.

Argument Block

E: 3
E+1: software status word, with the following status bits settable by the caller:

B0(MO%LCP) Set line printer as a lower case printer.

B12(MO%EOF) Set bit MO%EOF in the printer status word when all data sent to printer has actually been printed. The status word can be obtained with the .MORST function.

B14(MO%SER) Clear the software error condition on the line printer. This condition usually occurs on a character interrupt.

Other status bits can be read with the .MORST function (see below) but cannot be set by the caller.

E+2: value for page counter register. The caller can indicate the number of pages to be printed by specifying a value of up to 12 bits (4096). Each time the printer reaches the top of a new page, it decrements the value by one. When the value becomes zero, the printer sets status bit MO%LPC and generates an interrupt if the .MOPSI function was given previously.

If the caller specifies a value of 0 in the register, the system will maintain the page counter and will not generate an interrupt to the caller when the page counter becomes zero.

If the caller specifies a value of -1 in the register, the value will be ignored.

37 .MORST Read the status of the line printer. The status is obtained from the front end, and the caller is blocked until it receives the status.

Argument Block

E: 3
E+1: status word. The following bits are defined:

B0(MO%LCP) Line printer is a lower case printer. This bit is set only if a .MOSTS function declaring the printer lower case was executed previously.

B1(MO%RLD)	Front end has been reloaded. This bit is reset to zero the next time any I/O activity begins for the line printer.
B10(MO%FER)	A fatal hardware error occurred. This condition generates a software interrupt on channel 11 (.ICDAE).
B12(MO%EOF)	All data sent to printer has actually been printed.
B13(MO%IOP)	Output to the line printer is in progress.
B14(MO%SER)	A software error (e.g., interrupt character, page counter overflow) occurred.
B15(MO%HE)	A hardware error occurred. This error generates a software interrupt on channel 11 (.ICDAE). This condition usually requires that the forms be realigned.
B16(MO%OL)	Line printer is off line. This bit is set on the occurrence of any hardware condition that requires operator intervention.
B17(MO%FNX)	Line printer does not exist.
B30(MO%RPE)	A RAM parity error occurred.
B31(MO%LVU)	The line printer has an optical (12-channel tape reader) VFU.
B33(MO%LVF)	A VFU error occurred. The paper has to be realigned.
B34(MO%LCI)	A character interrupt occurred. This generates a software interrupt on channel 11 (.ICDAE).
B35(MO%LPC)	The page counter register has overflowed.

Bits 2-17 contain the software status word from the front end, and bits 20-35 contain the hardware status word.

E+2: value of page counter register. A value of -1 indicates the printer has no page counter value defined.

40	.MOFLO	Flush any line printer output that has not yet been printed.

CDR Functions

The functions available for card readers (CDR) are described below. Like the LPT functions, these functions accept the address of an argument block in AC3. The first word of the block contains the length (including this word) of the block. Remaining words in the block contain arguments for the particular function.

Code	Symbol	Meaning
27	.MOPSI	Enable for a software interrupt on nonfatal device conditions. Examples of these conditions are: 1. Device changed from off line to on line. 2. Device changed from on line to off line. Other device errors or software conditions are not handled by this function; instead they cause a software interrupt on channel 11 (.ICDAE). Argument Block E: 3 E+1: interrupt channel number E+2: flags. The following flag is defined: B0(MO%MSG) Suppress standard CTY device messages.
37	.MORST	Read the status of the card reader. The status is obtained from the front end, and the caller is blocked until it receives the status. Argument Block E: 2 E+1: status word. Bits 2-17 contain the software status word from the front end, and bits 20-35 contain the hardware status word. B0(MO%COL) Card reader is on line. This bit is not obtained from the front end. B1(MO%RLD) Front end has been reloaded. This bit is reset to zero the next time I/O activity begins for the card reader. B10(MO%FER) A fatal hardware error occurred. This condition generates a software interrupt on channel 11 (.ICDAE). B12(MO%EOF) Card reader is at end of file. B13(MO%IOP) Input from the card reader is in progress. B14(MO%SER) A software error (e.g., interrupt character) occurred.

B15(MO%HE)	A hardware error occurred. This error generates a software interrupt on channel 11 (.ICDAE).
B16(MO%OL)	Card reader is off line. This bit is set on the occurrence of any hardware condition that requires operator intervention.
B17(MO%FNX)	Card reader does not exist.
B31(MO%SFL)	The output stacker is full.
B32(MO%HEM)	The input hopper is empty.
B33(MO%SCK)	A card did not stack correctly in the output stacker.
B34(MO%PCK)	The card reader failed to pick a card correctly from the input hopper.
B35(MO%RCK)	The card reader detected a read error when reading a card.

PTY Functions

The functions available for pseudo-terminals (PTY) are described below. Some of these functions accept arguments in AC3. (Refer to the individual descriptions.)

Code	Symbol	Meaning
24	.MOAPI	Assign PTY interrupt channels. This function requires AC2 to contain B0(MO%WFI) enable waiting-for-input interrupt B1(MO%OIR) enable output-is-ready interrupt B12-B17(MO%SIC) software interrupt channel number for output to the PTY. The channel number used for input from the PTY is one greater than the channel number used for output to the PTY. B18-B35 function code
25	.MOPIH	Determine if PTY job needs input. On a successful return, AC2 contains 0(.MONWI) if PTY job is not waiting for input or contains -1(.MOWFI) if PTY job is waiting for input.
26	.MOBAT	Set batch control bit. This function requires AC3 to contain 0(.MONCB) if the job is not to be controlled by batch or to contain 1(.MOJCB) if the job is to be controlled by batch. To obtain this value, the process can execute the GETJI JSYS, function .JIBAT.

FE Functions

Code	Symbol	Meaning
3	.MOEOF	Send an end of file to the program using the FE device on the front end. This function is used for synchronization between a program running on the DECSYSTEM-20 and a program running on the front end.
4	.MODTE	Assign the specified device to the DTE controller on the front end. This function, which must be performed before I/O is allowed to the device, requires AC3 to contain the device type. The process must have WHEEL or OPERATOR capability enabled.

TTY Functions

Code	Symbol	Meaning
25	.MOPIH	Determine if TTY job needs input. On a successful return, AC2 contains 0(.MONWI) if TTY job is not waiting for input or contains -1(.MOWFI) if TTY job is waiting for input.
26	.MOSPD	Set the terminal line speed. This function accepts in AC3 the desired line speed (input speed in the left half and output speed in the right half). The left half of AC2 contains flag bits indicating the type of line being set. If B0(MO%RMT) is on, the line is a remote (dataset) line. If B1(MO%AUT) is on, the line is a remote autobaud line (i.e., is automatically set at 300 baud, and the contents of AC3 are ignored. The process must have WHEEL or OPERATOR capability enabled to set B0(MO%RMT) and B1(MO%AUT).
27	.MORSP	Return the terminal line speed. On a successful return, the left half of AC2 contains flag bits indicating the type of line, and AC3 contains the speed (input speed in the left half and output speed in the right half). If B0(MO%RMT) of AC2 is on, the line is a remote line, and if B1(MO%AUT) is on, the line is a remote autobaud line. AC3 contains the speed or contains -1 if the speed is unknown or is not applicable.
30	.MORLW	Return the terminal page width. On a successful return, AC3 contains the width.
31	.MOSLW	Set the terminal page width. This function requires AC3 to contain the desired width.
32	.MORLL	Return the terminal page length. On a successful return, AC3 contains the length.
33	.MOSLL	Set the terminal page length. This function requires AC3 to contain the desired length.
34	.MOSNT	Specify if terminal line given in AC1 is to receive system messages. This function requires AC3 to contain 0 (.MOSMY) to allow messages or 1 (.MOSMN) to suppress messages.

35	.MORNT	Return a code indicating if terminal line given in AC1 is to receive system messages. On a successful return, AC3 contains 0 (.MOSMY) if messages are being sent to this line or 1 (.MOSMN) if messages are being suppressed to this line.
36	.MOSIG	Specify if input on this terminal line is to be ignored when the line is inactive (i.e., is not assigned or opened). This function requires AC3 to contain 0 if characters on this line are not to be ignored or 1 if characters on this line are to be ignored. When input is being ignored and characters are typed, no CTRL/G (bell) is sent, as is the normal case when characters are typed on an inactive line.

Generates an illegal instruction interrupt on error conditions below.

MTOPR ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX5:	file is not open
IOX5:	device or data error
MTOX1:	invalid function
MTOX2:	record size was not set before I/O was done
MTOX3:	function not legal in dump mode
MTOX4:	invalid record size
MTOX5:	invalid hardware data mode for magnetic tape
MTOX6:	invalid magnetic tape density
MTOX7:	WHEEL or OPERATOR capability required
MTOX8:	argument block too long
MTOX9:	output still pending
MTOX10:	VFU or RAM file cannot be OPENed
MTOX11:	data too large for buffers
MTOX12:	input error or not all data read
MTOX13:	argument block too small
MTOX14:	invalid software interrupt channel number
MTOX15:	device does not have Direct Access (programmable) VFU
MTOX16:	VFU or Translation RAM file must be on disk

MTOX17: device is not on line

MTOX18: invalid software interrupt channel number

MTOX19: invalid terminal line width

MTOX20: invalid terminal line length

TTYX01: line is not active

MUTIL JSYS 512

Performs various IPCF (Inter-Process Communication Facility) functions, such as enabling and disabling PIDs, assigning PIDs, and setting quotas. Refer to the DECSYSTEM-20 Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

ACCEPTS IN AC1: length of argument block

AC2: address of argument block

RETURNS +1: failure, error code in AC1

+2: success. Responses from the requested function are returned in the argument block.

The format of the argument block is as follows:

Word	Meaning
0	Code of desired function. (See below.)
1 through n	Arguments for the desired function. The arguments, which depend on the function requested, begin in word 1 and are given in the order shown below. Responses from the requested function are returned in these words.

The available functions, along with their arguments, are described below.

Code	Symbol	Meaning
1	.MUENB	Enable the specified PID to receive packets. If the caller is not the owner of the PID, the caller must have IPCF capability enabled. Argument PID
2	.MUDIS	Disable the specified PID from receiving packets. If the caller is not the owner of the PID, the caller must have IPCF capability enabled. Argument PID

3 .MUGTI Return the PID associated with <SYSTEM>INFO. The PID is returned in word 2 of the argument block.

Argument

PID or job number

4 .MUCPI Create a private copy of <SYSTEM>INFO for the specified job. The caller must have WHEEL, OPERATOR, or IPCF capability enabled.

Arguments

PID to be assigned to <SYSTEM>INFO
PID or number of job creating private copy

5 .MUDES Delete the specified PID. The caller must own the PID being deleted.

Argument

PID

6 .MUCRE Creates a PID for the specified process or job. The flags that can be specified are B6(IP%JWP) to make the PID job wide and B7(IP%NOA) to prevent access to PID from other processes. The caller must have IPCF capability enabled if the job number given is not that of the caller. The PID created is returned in word 2 of the argument block.

Argument

flags,,process handle or job number

7 .MUSSQ Set send and receive quotas for the specified PID. The caller must have IPCF capability enabled. The new send quota is given in B18-B26, and the new receive quota is given in B27-B35.

Argument

PID
new quotas

10 .MUCHO Change the job number associated with the specified PID. The caller must have WHEEL or OPERATOR capability enabled.

Arguments

PID
new job number or PID belonging to new job

11 .MUFOJ Return the job number associated with the specified PID. The job number is returned in word 2 of the argument block.

Argument

PID

12	.MUFJP	Return all PIDs associated with the specified job. Two words are returned, starting in word 2 of the argument block, for each PID. The first word is the PID. The second word has B6(IP%JWP) set if the PID is job wide and B7(IP%NOA) set if the PID is not accessible by other processes. The list is terminated by a 0 PID. Argument job number or PID belonging to that job
13	.MUFSQ	Return the send and receive quotas for the specified PID. The quotas are returned in word 2 of the argument block with the send quota in B18-B26 and the receive quota in B27-B35. Argument PID
15	.MUFFP	Return all PIDs associated with the same process as that of the specified PID. The list of PIDs returned is in the same format as the list returned for the .MUFJP function (12). Argument PID
16	.MUSPQ	Set the maximum number of PIDs allowed for the specified job. The caller must have IPCF capability enabled. Arguments job number or PID PID quota
17	.MUFPQ	Return the maximum number of PIDs allowed for the specified job. The PID quota is returned in word 2 of the argument block. Argument job number or PID
20	.MUQRY	Return the Packet Descriptor Block for the next packet in the queue associated with the specified PID. An argument of -1 returns the next descriptor block for the process, and an argument of -2 returns the next descriptor block for the job. The descriptor block is returned starting in word 1 of the argument block. Argument PID

21	.MUAPF	Associate the PID with the specified process. Arguments PID process handle
22	.MUPIC	Place the specified PID on a software interrupt channel. An interrupt is then generated when a message is received in the input queue associated with the PID. If the channel number is given as -1, the PID is removed from its current channel. Arguments PID channel number
23	.MUDFI	Set the PID of <SYSTEM>INFO. An error is given if <SYSTEM>INFO already has a PID. The caller must have IPCF capability enabled. Argument PID of <SYSTEM>INFO
24	.MUSSP	Place the specified PID into the system PID table at the given offset. The caller must have WHEEL, OPERATOR, or IPCF capability enabled. Arguments index into system PID table PID
25	.MURSP	Return a PID from the system PID table. The PID is returned in word 2 of the argument block. The system PID table currently has the following entries: 0 .SPIPC Reserved for DEC 1 .SPINF PID of <SYSTEM>INFO 2 .SPQSR PID of QUASAR 3 .SPMDA PID of QSRMDA Argument index into system PID table
26	.MUMPS	Return the maximum packet size for the specified PID. The size is returned in word 2 of the argument block. Argument PID

MUTIL ERROR MNEMONICS:

IPCFX2: no message for this PID

IPCFX3: data too long for user's buffer

IPCFX4:	receiver's PID invalid
IPCFX5:	receiver's PID disabled
IPCFX6:	send quota exceeded
IPCFX7:	receiver quota exceeded
IPCFX8:	IPCF free space exhausted
IPCFX9:	sender's PID invalid
IPCF10:	WHEEL capability required
IPCF11:	WHEEL or IPCF capability required
IPCF12:	no free PID's available
IPCF13:	PID quota exceeded
IPCF14:	no PID's available to this job
IPCF15:	no PID's available to this process
IPCF16:	receive and message data modes do not match
IPCF17:	argument block too small
IPCF18:	invalid MUTIL JSYS function
IPCF19:	no PID for [SYSTEM]INFO
IPCF20:	invalid process handle
IPCF21:	invalid job number
IPCF22:	invalid software interrupt channel number
IPCF23:	[SYSTEM]INFO already exists
IPCF24:	invalid message size
IPCF25:	PID does not belong to this job
IPCF26:	PID does not belong to this process
IPCF27:	PID is not defined
IPCF28:	PID not accessible by this process
IPCF29:	PID already being used by another process
IPCF30:	job is not logged in
IPCF32:	page is not private
IPCF33:	invalid index into system PID table
IPCF35:	invalid IPCF quota

NIN JSYS 225

Inputs an integer number, with leading spaces ignored. This call terminates on the first character not in the specified radix. If that character is a carriage return followed by a line feed, the line feed is also input.

ACCEPTS IN AC1: source designator

AC3: radix (2-10) of number being input

RETURNS +1: failure, error code in AC3, updated string pointer, if pertinent, in AC1

+2: success, number in AC2 and updated string pointer, if pertinent, in AC1

NIN ERROR MNEMONICS:

IFIXX1: radix is not in range 2 to 10

IFIXX2: first nonspace character is not a digit

IFIXX3: overflow (number is greater than 2**35)

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

NOUT JSYS 224

Outputs an integer number.

ACCEPTS IN AC1: destination designator

AC2: number to be output

AC3: B0(NO%MAG) output the magnitude. That is, output the number as an unsigned 36-bit number (e.g., output -1 as 777777 777777).

B1(NO%SGN) output a plus sign for a positive number.

B2(NO%LFL) output leading filler. If this bit is not set, trailing filler is output, and bit 3(NO%ZRO) is ignored.

B3(NO%ZRO) output 0's as the leading filler if the specified number of columns (NO%COL) allows filling. If this bit is not set, blanks are output as leading filler if the number of columns allows filling.

B4(NO%OOV)	output on column overflow and return an error. If this bit is not set, column overflow is not output.
B5(NO%AST)	output asterisks on column overflow. If this bit is not set and bit 4 (NO%OOV) is set, all necessary digits are output on column overflow.
B11-B17 (NO%COL)	number of columns (including sign column) to output. If this field is 0, as many columns as necessary are output.
B18-B35 (NO%RDX)	radix (2-36) of number being output

RETURNS +1: failure, error code in AC3

+2: success, updated string pointer in AC1, if pertinent

NOUT ERROR MNEMONICS:

NOUTX1: radix is not in range 2 to 36

NOUTX2: column overflow

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

IOX11: quota exceeded or disk full

ODCNV JSYS 222

Converts the internal date and time format into separate numbers for local weekday, day, month, year, and time and does not convert the numbers to text. (Refer to Section 2.8.2 for more information.) The ODCNV call gives the caller the option of explicitly specifying the time zone and daylight savings time.

ACCEPTS IN AC2: internal date and time, or -1 for current date and time

AC4:

B0(IC%DSA)	apply daylight savings according to the setting of B1(IC%ADS). If B0 is off, daylight savings is applied only if appropriate for date.
B1(IC%ADS)	apply daylight savings if B0(IC%DSA) is on.
B2(IC%UTZ)	use time zone in B12-B17(IC%TMZ). If this bit is off, the local time zone is used.

B3(IC%JUD)	apply Julian day format (Jan 1 is day 1 in conversion)
B12-B17 (IC%TMZ)	time zone to use if B2(IC%UTZ) is on.

RETURNS +1: always, with

AC2 containing the year in the left half, and the numerical month (0=January) in the right half.

AC3 containing the day of the month (0=first day) in the left half, and the day of the week (0=Monday) in the right half.

AC4 containing

B0 and B2	on for compatibility with the IDCNV call
B1(IC%ADS)	on if daylight savings was applied
B3(IC%JUD)	on if Julian day format was applied
B12-B17 (IC%TMZ)	time zone used
B18-B35 (IC%TIM)	local time in seconds since midnight

Generates an illegal instruction interrupt on error conditions below.

ODCNV ERROR MNEMONICS:

DATEX6: system date and time are not set

TIMEX1: time cannot be greater than 24 hours

ZONEX1: time zone out of range

ODTIM JSYS 220

Outputs the date and time by converting the internal format of the date and/or time to text. (Refer to Section 2.8.2.)

ACCEPTS IN AC1: destination designator

AC2: internal date and time, or -1 for current date and time

AC3: format option flags (see below), 0 is the normal case

RETURNS +1: always, updated string pointer in AC1, if pertinent

The format option flags in AC3 indicate the format in which the date and time are to be output.

ODTIM Option Flags

B0(OT%NDA)	do not output the date and ignore B1-B8.
B1(OT%DAY)	output the day of the week according to the format specified by B2(OT%FDY).
B2(OT%FDY)	output the full text for the day of the week. If this bit is off, the 3-letter abbreviation of the day of the week is output.
B3(OT%NMN)	output the month as numeric and ignore B4(OT%FMN).
B4(OT%FMN)	output the full text for the month. If this bit is off, the 3-letter abbreviation of the month is output.
B5(OT%4YR)	output the year as a 4-digit number. If this bit is off, the year is output as a 2-digit number if between 1900 and 1999.
B6(OT%DAM)	output the day of the month after the month. If this bit is off, the day is output before the month.
B7(OT%SPA)	output the date with spaces between the items (e.g., 6 Feb 76). If B6(OT%DAM) is also on, a comma is output after the day of the month (e.g., Feb 6, 76).
B8(OT%SLA)	output the date with slashes (e.g., 2/6/76).
	If B7-B8 are both off, the date is output with dashes between the items (e.g., 6-Feb-76).
B9(OT%NTM)	do not output the time and ignore B10-B13.
B10(OT%NSC)	do not output the seconds. If this bit is off, the seconds are output, preceded by a colon.
B11(OT%12H)	output the time in 12-hour format with AM or PM following the time. If this bit is off, the time is output in 24-hour format.
B12(OT%NCO)	output the time without a colon between the hours and minutes.
B13(OT%TMZ)	output the time and follow it with a "-" and a time zone (e.g., -EDT).
B17(OT%SCL)	suppress columnation of the date and time by omitting leading spaces and zeros. This produces appropriate output for a message. If this bit is off, the date and time are output in columns of constant width regardless of the particular date or time. However, full texts of months and weekdays are not columnated. This output is appropriate for tables.

If AC3 is 0, the ODTIM call outputs the date and time in columns in the format

```
dd-mmm-yy  hh:mm:ss
```

For example, 6-Feb-76 15:14:03.

If AC3 is -1, the ODTIM call interprets the contents as if B1-B2,B4-B7, and B17 were on (i.e., AC3=336001000000) and outputs the date and time in the format

weekday, month day, year hh:mm:ss

as in Friday, February 6, 1976 15:14:03

Additional examples are:

Contents of AC3	Typical Text
202201000000	Fri 6 Feb 76 1:06
336321000000	Friday, February 6, 1976 1:06AM-EST
041041000000	6/2/76 106:03
041040000000	6/02/76 106:03

Generates an illegal instruction interrupt on error conditions below.

ODTIM ERROR MNEMONICS:

DATEX6: system date and time are not set

TIMEX1: time cannot be greater than 24 hours

All I/O errors are also possible. These errors cause software interrupts or process terminations as described for the BOUT call description.

ODTNC JSYS 230

Outputs the date and/or the time as separate numbers for local year, month, day, or time. (Refer to Section 2.8.2.) This JSYS is a subset of the ODTIM call because the output of dates and times not stored in internal format is permitted. Also, the caller has control over the time and zone printed.

ACCEPTS IN AC1: destination designator

AC2: year in the left half, and numerical month (0=January) in the right half

AC3: day of the month (0=first day) in the left half, and day of the week (0=Monday), if desired, in the right half

AC4:

B1(IC%ADS)	apply daylight savings on output
B3(IC%JUD)	apply Julian day format on output
B12-B17(IC%TMZ)	time zone in which to output
B18-B35(IC%TIM)	local time in seconds since midnight

AC5: format option flags (refer to ODTIM for the description of these flags)

NOTE

The only time zones that can be output by B13(OT%TMZ) are Greenwich and USA zones.

RETURNS +1: always, updated string pointer in AC1, if pertinent.

Generates an illegal instruction interrupt on error conditions below.

ODTNC ERROR MNEMONICS:

DATEX1: year out of range

DATEX2: month is not less than 12

DATEX3: day of month too large

DATEX4: day of week is not less than 7

ZONEX1: time zone out or range

ODTNX1: time zone must be USA or Greenwich

All I/O errors can occur. These errors cause software interrupts or process terminations as described for the BOUT call description.

OPENF JSYS 21

Opens the given file. Refer to the DECSYSTEM-20 Monitor Calls User's Guide for the explanations of the types of access allowed to a file.

ACCEPTS IN AC1: JFN in the right half of the file being opened.

AC2: B0-B5(OF%BSZ) Byte size (maximum of 36 decimal).

B6-B9(OF%MOD) File data mode. A value of 0 indicates normal data mode for the device (dump mode is illegal). A value of 17 indicates dump mode (byte I/O is illegal and byte size is ignored). A value of 10 indicates image mode, a value of 13 indicates image binary mode, and a value of 14 indicates binary mode.

B18(OF%HER)	Halt on I/O device or data error. If this bit is on and a condition occurs that causes an I/O device or data error interrupt, the process will instead be halted, and an illegal instruction interrupt will be generated. If this bit is off and the condition occurs, the interrupt is generated on its normally-assigned channel.
B19(OF%RD)	Allow read access.
B20(OF%WR)	Allow write access.
B21(OF%EX)	Reserved for future use.
B22(OF%APP)	Allow append access.
B25(OF%THW)	Allow thawed access. If this bit is off, the file is opened for frozen access.
B26(OF%AWT)	Block program and print a message on the job's controlling terminal if access to file cannot be permitted. The program is blocked until access is granted.
B27(OF%PDT)	Do not update access dates of the file.
B28(OF%NWT)	Return an error if access to file cannot be permitted. If B26 and B28 are both off, the default is to return an error if access to the file cannot be granted.
B29(OF%RTD)	Allow restricted access (i.e., allow only one process to access the file).
B30(OF%PLN)	Disable line number checking and consider a line number as 5 characters of text.
B31(OF%DUD)	Suppress the system updating of modified pages in memory to thawed files on the disk.
B32(OF%OFL)	Open the device even if it is off line.

RETURNS +1: failure, error code in AC1

+2: success

A byte size of 0 opens the file in 36-bit (full word) bytes and can be given when subsequent I/O will be done with the PMAP call. A byte size of 8 given for a JFN referring to a terminal opens the terminal in binary mode. (Refer to Section 2.4.3.) A byte size of 8 given for a JFN referring to a line printer opens the printer in 8-bit byte mode, thereby allowing the data to be accessed with 8-bit byte pointers. (Refer to Section 2.4.1.) A byte size other than 8 given for the printer results in using 7-bit bytes.

NOTE

For sequential I/O, bytes are packed into a 36-bit monitor buffer and are unpacked by a pointer standard for the device and data mode being used. For example, a byte size of 7 (which is the system standard for ASCII data) causes 5 7-bit bytes to be packed into the buffer with the setting of the remaining bit being unpredictable. If a byte size is then used that retrieves all 36 bits of data from the buffer, the last bit of each word is indeterminate.

Even though each type of desired access can be indicated by a separate bit, some accesses are implied when specific bits are set. For example, the setting of the write access bit implies read access if the process is allowed to read the file according to the file's access code. This means that if the process has access to read the file and it sets only the write access bit, the process will have the file opened for read, write, and execute access. However, if an existing file is opened and only write access is specified (only OF%WR is set), the contents of the file are deleted, and the file is considered empty. Thus, to update an existing file, both OF%RD and OF%WR must be set.

The CLOSF monitor call can be used to close a specific file.

OPENF ERROR MNEMONICS:

OPNX1: file is already open

OPNX2: file does not exist

OPNX3: read access required

OPNX4: write access required

OPNX5: execute access required

OPNX6: append access required

OPNX7: device already assigned to another job

OPNX8: device is not on line

OPNX9: invalid simultaneous access

OPNX10: entire file structure full

OPNX12: list access required

OPNX13: invalid access requested

OPNX14: invalid mode requested

OPNX15: read/write access required

OPNX16: file has bad index block

OPNX17: no room in job for long file page table

OPNX18: Unit Record Devices are not available

OPNX23: disk quota exceeded

OPNX25: device is write-locked

OPNX26: illegal to open a string pointer

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

SFBSX2: invalid byte size

TTYX01: line is not active

PBIN JSYS 73

Inputs the next sequential byte from the primary input designator. This call is equivalent to a BIN call with the source designator given as .PRIIN.

RETURNS +1: always, with the byte right-justified in AC1

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

PBIN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX5: file is not open

IOX1: file is not open for reading

IOX4: end of file reached

IOX5: device or data error

PBOUT JSYS 74

Outputs a byte sequentially to the primary output designator. This call is equivalent to a BOUT call with the destination designator given as .PRIOU.

ACCEPTS IN AC1: byte to be output, right-justified

RETURNS +1: always

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

PBOUT ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

IOX2: file is not open for writing

IOX5: device or data error

IOX6: illegal to write beyond absolute end of file

IOX11: quota exceeded or disk full

PEEK JSYS 311

Transfers a block of words from the monitor to the user space. The desired monitor pages must have read access. This monitor call is used to obtain data from the monitor for maintenance and test purposes and should be executed only when GETAB information is not available.

ACCEPTS IN AC1: word count in the left half, and first virtual address of the monitor in the right half

AC2: first user address

RETURNS +1: failure, error code in AC1

+2: success, the desired words are transferred.

The PEEK monitor call requires the process to have the MAINTENANCE, WHEEL, or OPERATOR capability enabled.

PEEK ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required

PEEKX2: read access failure on monitor page

PMAP JSYS 56

Maps one or more complete pages from a file to a process (for input), from a process to a file (for output), or from one process to another process. Each of the three uses of PMAP is described below.

Case I Mapping File Pages to a Process

This use of the PMAP call does not actually transfer any data; it simply changes the contents of the process' page map. When changes are made to the page in the process, the changes will also be reflected in the page in the file, if write access has been specified for the file.

ACCEPTS IN AC1: JFN of the file in the left half, and the page number in the file in the right half. This AC contains the source.

AC2: process handle in the left half, and the page number in the process in the right half. This AC contains the destination.

AC3:

B0(PM%CNT)	A count is in the right half of AC3. This count specifies the number of sequential pages to be mapped.
B2(PM%RD)	Permit read access to the page.
B3(PM%WR)	Permit write access to the page.
B4(PM%EX)	Reserved for future use. The symbol PM%RWX can be used to set B2-B4.
B5(PM%PLD)	Preload the page being mapped (i.e., move the page immediately instead of waiting until it is referenced).
B9(PM%CPY)	Create a private copy of the page when it is written into (copy-on-write). If the page is mapped between two processes (Case III below), both processes will receive a private copy of the page.
B18-B35 (PM%RPT)	Number of pages to be mapped if B0(PM%CNT) is set.

RETURNS +1: always

This use of PMAP changes the map of the process such that addresses in the page in the process specified by the right half of AC2 actually refer to the page in the file specified by the right half of AC1. The present contents of the page in the process are removed. If the page in the file is currently nonexistent, it will be created when it is written (i.e., when the corresponding page in the process is written).

This use of PMAP is legal only if the file is opened for at least read access. The access bits specified in the PMAP call are ANDed with the access that was specified when the file was opened. However, copy-on-write is always granted, regardless of the file's access. The access granted is placed in the process' map.

The file cannot be closed while any of its pages are mapped into any process. Thus, before the file is closed, pages must be unmapped from each process by a PMAP call with -1 in AC1 (see below).

Case II Mapping Process Pages to a File

This use of the PMAP call actually transfers data by moving the contents of the specified page in the process to the specified page in the file. The process' map for that page becomes empty.

ACCEPTS IN AC1: process handle in the left half, and the page number in the process in the right half. This AC contains the source.

AC2: JFN of the file in the left half, and the page number in the file in the right half. This AC contains the destination.

AC3: access bits and repetition count. (Refer to Case I.)

RETURNS +1: always

The process page and the file page must be private pages. The ownership of the process page is transferred to the file page. The present contents of the page in the file is deleted.

The access granted to the file page is determined by ANDing the access specified in the PMAP call with the access specified when the file was opened.

When mapping pages from a process to a file, the end-of-file byte pointer and the byte size are not automatically updated in the File Descriptor Block (FDB). To allow the file to be read later via the sequential I/O calls (e.g., BIN, SIN), the process executing the PMAP call should close the file keeping the JFN (CLOSF call, bit CO%NRJ), update the byte pointer and the byte size in the FDB (CHFDB call), and then release the JFN (RLJFN call). (Refer to Section 2.2.8 for the format of the FDB fields.)

Case III Mapping One Process' Pages to Another Process

This use of the PMAP call normally does not transfer any data; it simply changes the contents of the page maps of the processes. When changes are made to the page in one process, the changes will also be reflected in the corresponding page in the other process.

ACCEPTS IN AC1: process handle in the left half, and the page number in the process in the right half. This AC contains the source.

AC2: a second process handle in the left half, and page number in that process in the right half. This AC contains the destination.

AC3: access bits and repetition count. (Refer to Case I.)

RETURNS +1: always

This use of PMAP changes the map of the destination process such that addresses in the page specified by the right half of AC2 actually refer to the page in the source process specified by the right half of AC1. The present contents of the destination page are deleted.

The access granted to the destination page is determined by the access specified in the PMAP call.

Unmapping Pages In a Process

As stated previously, a file cannot be closed if any of its pages are mapped in any process. To unmap the file's pages from a process, a PMAP call is executed with

AC1:	-1	
AC2:	process handle in the left half, and page number in the process in the right half	
AC3:	B0(PM%CNT)	Repeat count. Only the process page numbers are incremented.
	B18-B35	Number of pages to remove from process

This format of the PMAP call removes the pages indicated in AC2 from the process.

Illegal PMAP calls

The PMAP call is illegal if:

1. Both AC1 and AC2 designate files.
2. Both AC1 and AC2 are 0.
3. The PMAP call designates a file with write-only access.
4. The PMAP call designates a file with append-only access.

Can cause several software interrupts on certain file conditions.

Generates an illegal instruction interrupt on error conditions below.

PMAP ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX3:	JFN is not assigned
DESX5:	file is not open
DESX7:	JFN cannot refer to output wildcard designators
PMAPX1:	invalid access requested
PMAPX2:	invalid use of PMAP
PMAPX3:	illegal to move shared page into file
PMAPX4:	illegal to move file page into process
PMAPX5:	illegal to move special page into file

PMAPX6:	disk quota exceeded
PMAPX7:	illegal to map file on dismounted structure
FRKHX1:	invalid process handle
FRKHX2:	illegal to manipulate a superior process
FRKHX3:	invalid use of multiple process handle
FRKHX7:	process page cannot exceed 777
LNGFX1:	page table does not exist and file not open for write
IOX11:	quota exceeded or disk full
ARGX06:	invalid page number

PMCTL JSYS 560

Controls physical memory. This call allows a privileged program to add or remove portions of physical memory and to control use of cache memory.

The PMCTL monitor call requires the process to have WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: function code

AC2: length of the argument block

AC3: address of the argument block

RETURNS +1: always

The defined functions and their argument blocks are as follows:

Function	Symbol	Meaning
0	.MCRCE	Return the status of cache memory. The status is returned in word .MCCST of the argument block. Argument Block 0 .MCCST If B35(MC%CEN) is on, the cache is enabled.
1	.MCSCE	Set the status of cache memory. Argument Block 0 .MCCST Enable the cache if B35(MC%CEN) is on.

2	.MCRPS	Return the status of the given page. The number of the page is given in word .MCPPN, and its status is returned in word .MCPST.				
		Argument Block				
		0	.MCPPN	Number of physical page		
		1	.MCPST	Returned page status. The status is represented by one of the following values:		
				0	.MCPSA	Page is available for normal use.
				1	.MCPSS	Page is in a transition state.
				2	.MCPSO	Page is off line because it is nonexistent. Nonexistent memory is marked as off line at system startup.
				3	.MCPSE	Page is off line because the monitor detected an error.
3	.MCSPS	Set the status of the given page. The number of the page is given in word .MCPPN, and the status value is given in word .MCPST.				
		Argument Block				
		0	.MCPPN	Number of physical page.		
		1	.MCPST	Status for page. The status is represented by one of the following values:		
				0	.MCPSA	Mark page available for normal use.
				2	.MCPSO	Mark page off line because it does not exist.
				3	.MCPSE	Mark page off line because it has an error.

Generates an illegal instruction interrupt on error conditions below.

PMCTL ERROR MNEMONICS:

CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required

PMCLX1: invalid page state or state transition

PMCLX2: requested physical page is unavailable

PMCLX3: requested physical page contains errors

ARGX02: invalid function

ARGX06: invalid page number

PPNST JSYS 557

Translates a project-programmer number (a TOPS-10 36-bit directory designator) to its corresponding string. The string consists of the structure name and a colon followed by the directory name enclosed in brackets. This monitor call and the STPPN monitor call should appear only in programs that require translations of project-programmer numbers. Both calls are temporary calls and may not be defined in future releases.

ACCEPTS IN AC1: destination designator

AC2: project-programmer number (36 bits)

AC3: pointer to structure name string for which the given project-programmer number applies

RETURNS +1: always, string written to destination, with updated string pointer, if pertinent, in AC1

If the structure name string is a logical name, then the first structure appearing in the logical name definition is used.

Generates an illegal instruction interrupt on error conditions below.

PPNST ERROR MNEMONICS:

PPNX1: invalid PPN

PPNX2: structure is not mounted

GJFX22: insufficient system resources (Job Storage Block full)

STDVX1: no such device

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

DELFX6: internal format of directory is incorrect

DIRX1: invalid directory number

DIRX2: insufficient system resources

DIRX3: internal format of directory is incorrect

STRX01: structure is not mounted

STRX06: no such user number

IOX11: quota exceeded or disk full

PRARG JSYS 545

Returns and/or sets arguments for the specified process.

ACCEPTS IN AC1: function code in the left half, and a process handle in the right half

AC2: address of argument block

AC3: length of argument block

RETURNS +1: always, with the number of words in the argument block in AC3

The codes for the functions are as follows:

1	.PRARD	return the arguments beginning at the address specified in AC2
2	.PRAST	set the arguments using the argument block at the address specified in AC2

Generates an illegal instruction interrupt on error conditions below.

PRARG ERROR MNEMONICS:

PRAX1: invalid PRARG function code

PRAX2: no room in monitor data base for argument block

PRAX3: PRARG argument block too large

PSOUT JSYS 76

Outputs a string sequentially to the primary output designator.

ACCEPTS IN AC1: pointer to an ASCIZ string in the caller's address space

RETURNS +1: always, updated string pointer in AC1

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

PSOUT ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

IOX2: file is not open for writing

IOX5: device or data error

IOX6: illegal to write beyond absolute end of file

IOX11: quota exceeded or disk full

RCDIR JSYS 553

Translates the given directory string to its corresponding 36-bit directory number. The directory string consists of the structure name or logical name and a colon followed by the directory name enclosed in either square brackets or angle brackets. No spaces can appear between the structure name and the directory name, and each field given must include its punctuation. An example of a directory string is PS:<SMITH>. If the structure name is omitted from the string, the user's connected structure is used. If the directory name is omitted from the string, the user's connected directory is used.

Recognition can be used on the string but only on the directory name field; recognition cannot be used on the structure name field. Partial recognition can be allowed so that a user can employ recognition when typing the name of a subdirectory. When recognition is used on the directory name field and the directory name is not ambiguous, the closing bracket is not required.

The directory name field can contain wildcard characters, and repeated RCDIR calls can be executed to obtain the numbers of the directories whose characters match the given directory. After the first call, each subsequent RCDIR call returns the number of the next directory in the group.

ACCEPTS IN AC1: flag bits in the left half

AC2: pointer to ASCIZ string to be translated, a JFN, a 36-bit user number, or a 36-bit directory number (given for the purpose of checking its validity)

AC3: 36-bit directory number (given when stepping to the next directory in a group of directories)

RETURNS +1: always, with

AC1 containing flag bits in the left half

AC2 containing an updated string pointer (if a pointer was supplied as the argument). If recognition was used, this pointer reflects the remainder of the string that was appended to the original string.

AC3 containing a 36-bit directory number if execution of the call was successful

The flag bits supplied in the left half of AC1 are as follows:

B14(RC%PAR)	Allow partial recognition on the directory name. If the name given matches more than one directory, bit RC%AMB is set on return and the string is updated to reflect the unique portion of the directory name. If bit RC%PAR is not set, the name given matches more than one directory, and recognition is being used, bit RC%AMB is set on return, but the string is not updated.
B15(RC%STP)	Step to the next directory in the group and return the number of that directory. AC1 must have bit RC%AWL set. AC2 must contain a pointer to a string that contains wildcard characters in the directory name field. AC3 must contain a directory number.
B16(RC%AWL)	Allow the directory name to contain wildcard characters. No recognition is performed on a directory name that contains wildcard characters. Also, the directory name must include its terminating bracket. This bit must be set if bit RC%STP is also set.
B17(RC%EMO)	Match the given string exactly. When both the RC%PAR and RC%EMO bits are on, recognition is not used on the string, and the string is matched exactly. If this bit is off, recognition is used on the string.

The flag bits returned in the left half of AC1 are as follows:

On success

B0(RC%DIR)	Directory can be used only by connecting to it (i.e., it is a files-only directory). If this bit is off, the user can also login to (if the directory is on the public structure) or access this directory.
B1(RC%ANA)	Alphanumeric accounts can be used for this directory. If this bit is off, only numeric accounts can be used.
B2(RC%RLM)	All messages from <SYSTEM>MAIL.TXT are repeated every time the user logs in. If this bit is off, messages are printed only once.
B6(RC%WLD)	The directory name given contained wildcard characters.

On failure

B3(RC%NOM) — No match was found for the string given. This bit is returned if either 1) bit RC%EMO was on in the call and a string was given that matched more than one directory or 2) the syntax of the fields in the string is correct but the structure is not mounted or the directory does not exist.

B4(RC%AMB) — The argument given was ambiguous. This bit is returned if bit RC%EMO was off in the call and the string given either matched more than one directory or did not include the beginning bracket of the directory name field.

B5(RC%NMD) — There are no more directories in the group of directories. This bit is returned if RC%STP was on in the call and the numbers of all the directories in the group have been returned.

The RCDIR monitor call can be used in one of two ways. The simplest way is to translate a directory string that corresponds to only one directory to its corresponding 36-bit directory number. The string can be either recognized or matched exactly. Instead of accepting a string, the program can supply a JFN or a 36-bit user number and translate this argument to a directory number. When a JFN is supplied as an argument, the number returned is that of the directory containing the file associated with the JFN. When a user number is supplied as an argument, the number returned is the logged-in directory for that user. Finally, the program can supply a directory number to check the number's validity, and if the RCDIR call is successful, this same number is returned.

The second way of using the RCDIR call is to accept a directory string that corresponds to more than one directory and to step through all the directories matching the given string to obtain all the directory numbers. Repeated RCDIR calls are executed until the number of the last directory is returned. This use of RCDIR requires AC2 to contain a pointer to a string containing wildcard characters and is ignored if the string does not contain wildcard characters or if any other argument is given in AC2.

The first RCDIR call executed must have bit RC%AWL set in AC1 and the pointer to the string in AC2. If execution of the call is successful, AC3 contains the number of the directory corresponding to the first directory in the group. For example, if the string given is <SMITH*> and the call is successful, the number returned corresponds to <SMITH>. Subsequent RCDIR calls must set bits RC%STP and RC%AWL in AC1, reset the pointer in AC2 (because it is updated on a successful RCDIR call), and leave in AC3 the directory number returned from the previous RCDIR call. (The directory number in AC3 is accepted only if RC%STP is set in AC1 and a pointer to a string containing wildcard characters is given in AC2.) On successful execution of each subsequent RCDIR call, the number returned in AC3 corresponds to the next directory in the group. When the number of the last directory in the group has been returned, a subsequent RCDIR call sets bit RC%NMD in AC1; the content of AC3 is indeterminate.

The RCUSR monitor call can be used to translate a user name string to its corresponding user number. The DIRST monitor call can be used to translate either a directory number or a user number to its corresponding string.

Generates an illegal instruction interrupt on error conditions below.

RCDIR ERROR MNEMONICS:

RCDIX1: insufficient system resources

RCDIX2: invalid directory specification

RCDIX3: invalid structure name

RCDIX4: monitor internal error

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

DESX8: file is not on disk

DESX10: structure is dismounted

STRX01: structure is not mounted

RCM JSYS 134

Returns the word mask of the activated interrupt channels for the specified process. (Refer to Section 2.5.1 and the AIC and DIC calls for information on activating and deactivating software interrupt channels.)

ACCEPTS IN AC1: process handle

RETURNS +1: always, 36-bit word in AC1, with bit n on meaning channel n is activated

Generates an illegal instruction interrupt on error conditions below.

RCM ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RCUSR JSYS 554

Translates the given user name string to its corresponding 36-bit user number. The user name string consists of the user's name without any punctuation. The string must be associated with a directory on structure PS: that is not a files-only directory.

Recognition can be used on the string. In addition, the string can contain wildcard characters.

ACCPETS IN AC1: flag bits in the left half

AC2: pointer to ASCII string to be translated

AC3: 36-bit user number (given when stepping to the next user name in a group)

RETURNS +1: always, with

AC1 containing flag bits in the left half

AC2 containing an updated string pointer. If recognition was used, this pointer reflects the remainder of the string that is appended to the original string.

AC3 containing a 36-bit user number if execution of the call was successful

The flag bits supplied in the left half of AC1 are as follows. For additional information on these bits, refer to the RCDIR monitor call description.

B14(RC%PAR)	Allow partial recognition on the user name string.
B15(RC%STP)	Step to the next user name in the group.
B16(RC%AWL)	Allow the user name to contain wildcard characters.
B17(RC%EMO)	Match the given string exactly.

The flag bits returned in the left half of AC1 are as follows. For additional information on these bits, refer to the RCDIR monitor call description.

On success

B1(RC%ANA)	User can use alphanumeric accounts. If this bit is off, the user can use only numeric accounts.
B2(RC%RLM)	User sees all messages from <SYSTEM>MAIL.TXT every time he logs in. If this bit is off, the user sees the messages only once.
B6(RC%WLD)	The user name given contained wildcard characters.

On failure

B3(RC%NOM)	No match was found for the string given. This bit will be on if the string given refers to a files-only directory, if there is no directory on PS: that is associated with the user name string, or bit RC%EMO was on in the call and a string was given that matched more than one user.
B4(RC%AMB)	The string given was ambiguous because it matched more than one user.
B5(RC%NMD)	There are no more user names in the group.

The RCDIR monitor call can be used to translate a directory string to its corresponding directory number. The DIRST monitor call can be used to translate either a user number or a directory number to its corresponding string.

Generates an illegal instruction interrupt on error conditions below.

RCUSR ERROR MNEMONICS:

RCUSX1: insufficient system resources

RCDIX4: monitor internal error

STRX07: invalid user number

STRX08: invalid user name

RDTTY JSYS 523

Reads input from the primary input designator (.PRIIN) into the caller's address space. Input is read until either a break character is encountered or the given byte count is exhausted, whichever occurs first. Output generated as a result of character editing is output to the primary output designator (.PRIOU).

The RDTTY call handles the following editing functions:

1. Delete the last character input (DELETE).

2. Delete back to the last punctuation character (CTRL/W).

3. Delete back to the beginning of the current line or, if the current line is empty, back to the beginning of the previous line (CTRL/U).

4. Retype the current line from its beginning or, if the current line is empty, retype the previous line (CTRL/R).

5. Accept the next character without regard to its usual meaning (CTRL/V).

By handling these functions, the RDTTY call serves as an interface between the terminal and the user program.

ACCEPTS IN AC1:	pointer to string in caller's address space where input is to be placed	
AC2:	B0(RD%BRK)	Break on CTRL/Z or ESC.
	B1(RD%TOP)	Break on CTRL/G, CTRL/L, CTRL/Z, ESC, carriage return, line feed.
	B2(RD%PUN)	Break on punctuation (see below).
	B3(RD%BEL)	Break on end of line (carriage return and line feed, or line feed only).
	B4(RD%CRF)	Suppress a carriage return and return a line feed only.
	B5(RD%RND)	Return to user program if user tries to delete beyond beginning of the input buffer (e.g., user types a CTRL/U or DELETE past the first character in the buffer). If this bit is not set, the call rings the terminal's bell and waits for more input.
	B7(RD%RIE)	Return to user program if input buffer is empty. If this bit is not set, the call waits for more input.
	B10(RD%RAI)	Convert lower-case input to upper-case input.
	B11(RD%SUI)	Suppress CTRL/U indication (i.e., do not print XXX, and on display terminals, do not delete the characters from the screen).
	B18-B35	Number of bytes available in the string. The input is terminated when this count is exhausted, even if the specified break character has not yet been typed.

If the left half of AC2 is 0, the input is terminated on end of line only.

AC3: pointer to prompting-text (CTRL/R) buffer, or 0 if no buffer. The text in this buffer, followed by any text in the input buffer, is output if the user types CTRL/R in his first line of input. If no prompting-text buffer exists or the user types CTRL/R on other than the first line of input, only the text on the current line will be output.

RETURNS +1: failure, error code in AC1

+2: success, updated string pointer in AC1, appropriate bits set in the left half of AC2, and updated count of available bytes in the right half of AC2

The bits returned in the left half of AC2 on a successful return are:

B12(RD%BTM)	Break character terminated the input. If this bit is not set, the input was terminated because the byte count was exhausted.
B13(RD%BFE)	Control was returned to the program because the user tried to delete beyond the beginning of the input buffer and RD%RND was on in the call.
B14(RD%BLR)	The backup limit for editing was reached.

NOTE

Bits not described are reserved for use by the monitor. The state of these bits on completion of the RDTTY call is undefined.

The punctuation break character set (RD%PUN) is as follows:

CTRL/A-CTRL/F	ASCII codes 34-36
CTRL/H-CTRL/I	ASCII codes 40-57
CTRL/K	ASCII codes 72-100
CTRL/N-CTRL/Q	ASCII codes 133-140
CTRL/S-CTRL/T	ASCII codes 173-176
CTRL/X-CTRL/Y	

Upon completion of the call, the terminating character is stored in the string, followed by a NULL (unless the byte count was exhausted). Also, any CTRL/V, along with the character following it, is stored in the string.

RDTTY ERROR MNEMONICS:

RDTX1: invalid string pointer

IOX11: quota exceeded or disk full

RELD JSYS 71

Releases one or all devices assigned to the job. When a device is released by the job, the resource allocator receives an IPCF packet. (Refer to the ALLOC monitor call description for the format of the packet sent to the allocator.)

ACCEPTS IN AC1: device designator, or -1 to release all devices assigned to this job

RETURNS +1: failure, error code in AC1

+2: success

The ASND monitor call can be used to assign a device to the caller.

RELD ERROR MNEMONICS:

DEVX1: invalid device designator

DEVX2: device already assigned to another job

RESET JSYS 147

Resets and initializes the current process. It is a good programming practice to include this call at the beginning of each assembly language program.

RETURNS +1: always

The RESET monitor call performs the following:

1. Closes all files at or below the current process and releases all JFNs. If a file is nonexistent (i.e., has never been closed), it is closed and then expunged.

2. Kills all inferior processes.

3. Clears the current process' software interrupt system. The channel table and priority level table addresses remain unchanged from any previous settings.

4. Sets the following fields of the controlling terminal's JFN mode word (refer to Section 2.4.3.1):

TT%WAK(B18-B23)	to wake up on every character
TT%ECO(B24)	to cause echoing
.TTASI(B29)	to translate both echo and output (ASCII data mode)

   Remaining fields of the mode word are not changed.

5. Releases all of the current process' PIDs.

6. Dequeues all of the current process' ENQ requests.

7. Clears the compatibility package's entry vector.

8. Releases all process handles that can be released. (Refer to the RFRKH call description.)

RFACS JSYS 161

Returns the ACs of the specified process.

ACCEPTS IN AC1: process handle

AC2: address of the beginning of a 20(octal) word table in the caller's address space where the AC values of the specified process are to be stored

RETURNS +1: always

The SFACS monitor call can be used to set the ACs for a specified process.

Generates an illegal instruction interrupt on error conditions below.

RFACS ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

FRKHX4: process is running

RFBSZ JSYS 45

Returns the byte size for a specific opening of a file. (Refer to the OPENF or SFBSZ call description for setting the byte size.)

ACCEPTS IN AC1: JFN

RETURNS +1: failure, error code in AC1

+2: success, byte size right-justified in AC2

RFBSZ ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

RFCOC JSYS 112

Returns the control character output control (CCOC) words for the specified terminal. (Refer to Section 2.4.3.2.)

ACCEPTS IN AC1: file designator

RETURNS +1: always, with output control words in AC2 and AC3

The CCOC words consist of 2-bit bytes, each byte representing the output control for one of the ASCII codes 0-37. If the given designator is not associated with a terminal, the CCOC words are returned in AC2 and AC3 with each 2-bit byte containing a value of 2 (send actual code and account format action).

The SFCOC monitor call can be used to set the CCOC words for a specified terminal.

Generates an illegal instruction interrupt on error conditions below.

RFCOC ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

RFMOD JSYS 107

Returns the JFN mode word associated with the specified file. (Refer to Section 2.4.3.1.) The MTOPR monitor call should be used to return the page length and width fields, especially when the fields have values greater than 127. The RFMOD call returns these fields as 1 when their values are greater than 127.

ACCEPTS IN AC1: source designator

RETURNS +1: always, with mode word in AC2

If the designator is not a terminal, the RFMOD call returns in AC2 a word in the following format

7B3+^D66B10+^D72B17+ 4 mode bits from the OPENF for the designator

This setting of the left half of AC2 indicates that the designator has mechanical form feed, mechanical tab, lower case, page length of 66, and page width of 72.

The SFMOD and STPAR monitor calls can be used to set various fields of the JFN mode word.

RFMOD ERROR MNEMONICS:

TTYX01: line is not active

RFORK JSYS 155

Resumes one or more processes that had been directly frozen. This monitor call does not resume a process that has been indirectly frozen. (Refer to Section 2.6.2.1.) Also, the RFORK call cannot be used to resume a process that is suspended because of a monitor call intercept. (Refer to the UTFRK call.)

ACCEPTS IN AC1: process handle

RETURNS +1: always

The RFORK monitor call is a no-op if the referenced process(s) was not directly frozen.

The FFORK monitor call can be used to freeze one or more processes.

Generates an illegal instruction interrupt on error conditions below.

RFORK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RFPOS JSYS 111

Returns the current position of the specified terminal's pointer. (Refer to Section 2.4.3.4 for information on page lengths and widths of terminals.)

ACCEPTS IN AC1: device designator

RETURNS +1: always, AC2 contains position within a page (i.e., line number) in the left half, and position within a line (i.e., column number) in the right half

AC2 contains 0 if the designator is not associated with a terminal.

The SFPOS monitor call can be used to set the position of the terminal's pointer.

Generates an illegal instruction interrupt on error conditions below.

RFPOS ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

RFPTR JSYS 43

Returns the current position of the specified file's pointer.

ACCEPTS IN AC1: JFN

RETURNS +1: failure, error code in AC1

+2: success, byte number in AC2

The SFPTR monitor call can be used to set the position of the file's pointer.

RFPTR ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

RFRKH JSYS 165

Releases the specified handle of a process. A handle can be released only if it describes either an existent process inferior to at least one other process in the job or a process that has been killed via KFORK (i.e., a nonexistent process).

ACCEPTS IN AC1: process handle, or -1 to release all relative handles that can be released

RETURNS +1: failure, error code in AC1

+2: success

The process handles released when AC1 is -1 are the ones released on a RESET or a KFORK monitor call.

RFRKH ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RFSTS JSYS 156

Returns the status of the specified process.

ACCEPTS IN AC1: process handle

RETURNS +1: always, with the status word in AC1 and the PC in AC2

The process status word has the following format:

B0(RF%FRZ) The process is frozen. If this bit is off, the process is not frozen.

B1-B17(RF%STS) The status code for the process. The following values are possible:

Value	Symbol	Meaning
0	.RFRUN	The process is runnable.
1	.RFIO	The process is dismissed for I/O.
2	.RFHLT	The process is dismissed by voluntary process termination (HFORK or HALTF) or was never started.
3	.RFFPT	The process is dismissed by forced process termination. Forced termination occurs when bit 17(SC%FRZ) of the process capability word is not set.
4	.RFWAT	The process is dismissed waiting for another process to terminate.
5	.RFSLP	The process is dismissed for a specified amount of time.
6	.RFTRP	The process is dismissed because it attempted to execute a call on which an intercept has been set by its superior (via the TFORK call).
7	.RFABK	The process is dismissed because it encountered an instruction on which an address break was set (via the ADBRK call).

B18-B35(RF%SIC) The number of the software interrupt channel that caused the forced process termination.

The RFSTS call returns with -1 in the left half of AC1 if the specified handle is assigned but refers to a deleted process. The call generates an illegal instruction interrupt if the handle is unassigned.

Generates an illegal instruction interrupt on error conditions below.

RFSTS ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RFTAD JSYS 533

Returns the dates and times associated with the specified file.

ACCEPTS IN AC1: source designator

AC2: address of argument block

AC3: length of argument block

RETURNS +1: always, dates returned in the argument block

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.RSWRT	Internal date and time file was last written.
1	.RSCRV	Internal date and time file was created.
2	.RSREF	Internal date and time file was last referenced.
3	.RSCRE	Internal system date and time of last write.

On a successful return, the values for the number of words specified in AC3 are returned in the argument block. Words in the argument block contain -1 if any one of the following occurs:

1. The corresponding date does not exist for the file.
2. The designator is not associated with a file.
3. The corresponding date is not currently assigned (i.e., the argument block contains more than 4 words).

The SFTAD monitor call can be used to set the dates and times associated with a specified file.

Generates an illegal instruction interrupt on error conditions below.

RFTAD ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX7: JFN cannot refer to output wildcard designators

RIN JSYS 54

Inputs a byte nonsequentially (i.e., random byte input) from the specified file. The size of the byte is that given in the OPENF call. The RIN call can be used only when reading data from disk files.

ACCEPTS IN AC1: JFN

AC3: byte number within the file

RETURNS +1: always, with the byte right-justified in AC2

If the end of the file is reached, AC2 contains 0. The program can process this end-of-file condition if an ERJMP or ERCAL is the next instruction following the RIN call. Upon successful execution of the call, the file's pointer is updated for subsequent I/O to the file.

The ROUT monitor call can be used to output a byte nonsequentially to a specified file.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

RIN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

IOX1: file is not open for reading

IOX3: illegal to change pointer for this opening of file

IOX4: end of file reached

IOX5: device or data error

RIR JSYS 144

Returns the channel and priority level table addresses for the specified process. (Refer to Section 2.5.3.) These table addresses are set by the SIR monitor call. The RIR monitor call is useful when several independent processes in one job want to share software interrupt tables.

ACCEPTS IN AC1: process handle

RETURNS +1: always, with the priority level table address in the left half of AC2, and the channel table address in the right half of AC2

AC2 contains 0 if the SIR monitor call has not been executed by the designated process.

Generates an illegal instruction interrupt on error conditions below.

RIR ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RIRCM JSYS 143

Returns the mask for reserved software interrupt channels for the specified process. A process is able to read its own or its inferiors' channel masks.

ACCEPTS IN AC1: process handle

RETURNS +1: always, with the reserved channel mask for the specified process in AC2

The SIRCM monitor call can be used to set the mask for reserved software interrupt channels.

Generates an illegal instruction interrupt on error conditions below.

RIRCM ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RLJFN JSYS 23

Releases the specified JFNs. A JFN cannot be released unless it either has never been opened or has already been closed. Also, a JFN cannot be released if it is currently being assigned by a process, unless that process is the same as the one executing the RLJFN and is not at interrupt level. The GS%ASG bit returned from a GTSTS call for the JFN indicates if the JFN is currently being assigned.

ACCEPTS IN AC1: JFN, or -1 to release all JFNs that do not specify open files

RETURNS +1: failure, error code in AC1

+2: success

RLJFN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

RJFNX1: file is not closed

RJFNX2: JFN is being used to accumulate filename

RJFNX3: JFN is not accessible by this process

OPNX1: file is already open

RMAP JSYS 61

Acquires a handle on a page in a process to determine the access allowed for that page.

ACCEPTS IN AC1: process handle in the left half, and a page number within the process in the right half

RETURNS +1: always, with a handle on the page in AC1, and access information in AC2. The handle in AC1 is a process/file designator in the left half and a page number in the right half.

The access information returned in AC2 is as follows:

B2(RM%RD)	read access allowed
B3(RM%WR)	write access allowed
B4(RM%EX)	execute access allowed
B5(RM%PEX)	page exists
B9(RM%CPY)	copy-on-write access

On rare conditions, if the specified page is shared with a file but no JFN is associated with the file, AC1 contains -1 and AC2 contains 0.

Generates an illegal instruction interrupt on error conditions below.

RMAP ERROR MNEMONICS:

FRKHX1: invalid process handle

RNAMF JSYS 35

Renames an existing file. The JFNs of both the existing file and the new file specification must be closed.

ACCEPTS IN AC1: JFN of existing file to be renamed (i.e., source file)

AC2: JFN of new file specification (i.e., destination file specification)

RETURNS +1: failure, error code in AC1

+2: success, JFN in AC1 is released, and the JFN in AC2 is associated with the file under its new file specification

If the JFN of the new file specification already refers to an existing file, the existing file's contents are expunged.

When a file is renamed, many of the attributes of the existing file are given to the renamed file. The settings of the following words in the FDB (refer to Section 2.2.8) are copied from the existing file to the renamed file.

```
Word      .FBCTL (FB%LNG, FB%DIR, FB%NOD, FB%BAT, FB%FCF)
Word      .FBADR
Word      .FBCRE
Word      .FBGEN (FB%DRN)
Word      .FBBYV (FB%BSZ, FB%MOD, FB%PGC)
Word      .FBSIZ
Word      .FBCRV
Word      .FBWRT
Word      .FBREF
Word      .FBCNT
Word      .FBUSW
```

Note that the setting of FB%PRM (permanent file) does not get copied. Thus, if a file with bit FB%PRM on is renamed, the renamed file has FB%PRM off. The existing file is left in a deleted state with its contents empty but its FDB existent.

RNAMF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

OPNX1:	file is already open
RNAMX1:	files are not on same device
RNAMX2:	destination file expunged
RNAMX3:	write or owner access to destination file required
RNAMX4:	quota exceeded in destination of rename
RNAMX5:	destination file is not closed
RNAMX6:	destination file has bad page table
RNAMX7:	source file expunged
RNAMX8:	write or owner access to source file required
RNAMX9:	source file is nonexistent
RNMX10:	source file is not closed
RNMX11:	source file has bad page table
RNMX12:	illegal to rename to self
RNMX13:	insufficient system resources

ROUT JSYS 55

Outputs a byte nonsequentially (i.e., random byte output) to the specified file. The size of the byte is that given in the OPENF call for the JFN. The ROUT call can be used only when writing data to disk files.

ACCEPTS IN	AC1:	JFN
	AC2:	the byte to be output, right-justified
	AC3:	the byte number within the file
RETURNS	+1:	always

Upon successful execution of the call, the file's pointer is updated for subsequent I/O to the file.

The RIN monitor call can be used to input a byte nonsequentially from a specified file.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

ROUT ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

IOX2: file is not opened for writing

IOX3: illegal to change pointer for this opening of file

IOX5: device or data error

IOX6: illegal to write beyond absolute end of file

IOX11: quota exceeded or disk full

RPACS JSYS 57

Returns the accessibility of a page.

ACCEPTS IN AC1: process/file designator in the left half, and page number within the file in the right half

RETURNS +1: always, with AC2 containing the following information:

B2(PA%RD)	read access allowed
B3(PA%WT)	write access allowed
B4(PA%EX)	execute access allowed
B5(PA%PEX)	page exists
B6(PA%IND)	indirect pointer
B9(PA%CPY)	copy-on-write
B10(PA%PRV)	private page
B20(P1%RD)	read access allowed in first pointer
B21(P1%WT)	write access allowed in first pointer
B22(P1%EX)	execute access allowed in first pointer
B23(P1%PEX)	page exists in first pointer
B27(P1%CPY)	copy-on-write in first pointer

The bits in the left half are the result of tracing any indirect pointer chains, and the bits in the right half contain information about the first pointer (the one in the map directly indicated by the argument) only. The left half and right half information will be different only if an indirect pointer was encountered in the first map. In this case, B6(PA%IND) is set, the left half access is less than or equal to the right half access, and B9(PA%CPY) is set if it was found set at any level. B5(PA%PEX) and B10(PA%PRV) always refer to the last pointer (i.e., first non-indirect pointer) encountered.

The SPACS monitor call can be used to set the accessibility of a page.

Generates an illegal instruction interrupt on error conditions below.

RPACS ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

DESX8: file is not on disk

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RPCAP JSYS 150

Returns the capabilities for the specified process. (Refer to Section 2.6.1 for the description of the capability word.)

ACCEPTS IN AC1: process handle

RETURNS +1: always, capabilities possible for this process in AC2, and capabilities enabled for this process in AC3

The EPCAP monitor call can be used to enable the capabilities of a process.

Generates an illegal instruction interrupt on error conditions below.

RPCAP ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX3: invalid use of multiple process handle

RSCAN JSYS 500

Places a text string in, or reads a text string from, the job's rescan buffer (an area of storage in the Job Storage Block). This facility allows a program to receive information that will be used as primary input for another program before this other program reads input from the terminal.

The RSCAN call has two steps: the acceptance and the use of the text string. Each step has a different calling sequence. The first step is to accept the text string to be used as input and to place this string in the rescan buffer. The calling sequence for this step specifies, in AC1, a pointer to the text string to be input. The second step is to cause the string to be available to the program for reading via the BIN call. The calling sequence for this second step specifies a function code of 0(.RSINI) in AC1 to indicate that the last string entered at command level from the terminal is available for reading. The program executing the RSCAN call can determine when the data has been read by issuing a function code of 1(.RSCNT), which returns the number of characters remaining in the buffer.

In other words, the first RSCAN call specifying a new text string stores the string in the rescan buffer but does not cause it to be read. A second RSCAN call must be given before the string can be read. This second RSCAN causes the system to provide input from the most recent string stored and can be given only once. After this second RSCAN, nothing will be read from the rescan buffer until another RSCAN call specifying a different text string is given. In addition, the job receives input from the rescan buffer only if the source for input in the BIN call is the JFN of the controlling terminal. Therefore, if the source for input is other than the controlling terminal, input will not come from the rescan buffer.

ACCEPTS IN AC1: pointer to a new text string, or 0 in the left half and function code in the right half

RETURNS +1: failure, error code in AC1

+2: success

The defined functions are as follows:

Function	Symbol	Meaning
0	.RSINI	Make the data in the buffer available as input to any process in the current job that is reading data from its controlling terminal.
1	.RSCNT	Return the number of characters remaining to be read in the buffer. This function does not cause data to be read; it is used to determine when all the data has been read after making the data available.

On a successful return, AC1 contains an updated string pointer if a pointer was given in the call. Otherwise, AC1 contains the number of characters in the rescan buffer or 0 if there are no characters.

RSCAN ERROR MNEMONICS:

RSCNX2: invalid function code

RTFRK JSYS 322

Returns the handle of the process that was suspended because of a monitor call intercept and the monitor call that the process was attempting to execute. The superior process monitoring the intercepts can receive only one interrupt at a time. Thus, the superior process should execute the RTFRK call after receiving an interrupt to identify the process that caused the interrupt.

The system maintains a queue of the processes that have been suspended and that are waiting to interrupt the superior process monitoring the intercepts. The RTFRK call advances the processes on the queue, and if the call is not executed, subsequent interrupts are not generated.

Refer to the TOPS-20AN Monitor Calls User's Guide for an overview and description of the monitor call intercept facility.

RETURNS +1: always, with AC1 containing the handle of the process that generated the interrupt, and AC2 containing the monitor call instruction that caused the process to be suspended. If no process is currently suspended because of a monitor call intercept, AC1 and AC2 contain 0 on return.

Because the process handle returned in AC1 is a relative process handle, it is possible that a process is currently suspended but that all relative handles are in use. In this case, the caller should release a relative process handle with the RFRKH call and then reissue the RTFRK call.

Generates an illegal instruction interrupt on error conditions below.

RTFRK ERROR MNEMONICS:

FRKHX6: all relative process handles in use

RTIW JSYS 173

Reads the terminal interrupt word (refer to Section 2.5.6) for the specified process or the entire job and returns the terminal interrupt word mask.

ACCEPTS IN AC1: B0(RT%DIM) return the mask for deferred terminal interrupts

B18-B35 (RT%PRH) process handle, or -5 for entire job

RETURNS +1: always, with the terminal interrupt mask in AC2, and the deferred terminal interrupt mask in AC3. The deferred interrupt mask is returned only if both B0(RT%DIM) is on and the right half of AC1 indicates a specific process.

The STIW monitor call can be used to set the terminal interrupt word masks.

Generates an illegal instruction interrupt on error conditions below.

RTIW ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RUNTM JSYS 15

Returns the runtime of the specified process or of the entire job.

ACCEPTS IN AC1: process handle, or -5 for the entire job

RETURNS +1: always, with runtime (in milliseconds) right-justified in AC1, a divisor to convert time to seconds in AC2, and console time (in milliseconds) in AC3. AC2 always contains 1000; thus, it is not necessary to examine its contents.

Generates an illegal instruction interrupt on error conditions below.

RUNTM ERROR MNEMONICS:

FRKHX1: invalid process handle

RUNTX1: invalid process handle -3 or -4

RWM JSYS 135

Returns the word mask for the interrupts waiting on software channels for the specified process.

ACCEPTS IN AC1: process handle

RETURNS +1: always, with

AC1 containing a 36-bit word with bit n on meaning an interrupt on channel n is waiting.

AC2 containing the status of the interrupts in progress. Bit n on in the left half means an interrupt of priority level n occurring during execution of user code is in progress. Bit 18+n on in the right half means an interrupt of priority level n occurring during execution of monitor code is in progress.

Generates an illegal instruction interrupt on error conditions below.

RWM ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

RWSET JSYS 176

Releases the working set by removing all of the current process' pages from its working set. The pages are moved to secondary storage and are not preloaded the next time the process is swapped in. This operation is invisible to the user.

RETURNS +1: always

SACTF JSYS 62

Sets the account to which the specified file is to be charged.

ACCEPTS IN AC1: JFN

AC2: account number in bits 3-35 if bits 0-2 are 5. Otherwise contains a pointer to an account string in the address space of caller. If a null byte is not seen, the string is terminated after 39 characters are processed.

RETURNS +1: failure, error code in AC1

+2: success, updated string pointer in AC2

If the account validation facility is enabled, the SACTF call verifies the account given and returns an error if it is not valid for the caller.

The GACTF monitor call can be used to obtain the account designator to which a file is being charged.

SACTF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

SACTX1: file is not on multiple-directory device

SACTX2: insufficient system resources (Job Storage Block full)

SACTX3: directory requires numeric account

SACTX4: write or owner access required

VACCX0: invalid account

VACCX1: account string exceeds 39 characters

VACCX2: account has expired

SAVE JSYS 202

Saves, in nonsharable format, pages of a process into the specified file. (Refer to Section 2.7.1 for the format of a nonsharable save file.) This file can then be copied into a given process with the GET monitor call.

ACCEPTS IN AC1: process handle in the left half, and JFN in the right half

AC2: one table entry, or 0 in the left half and pointer to the table in the right half (see below)

RETURNS +1: always

The table has words in the format: length of the area to save in the left half and address of the first word to save in the right half. The table is terminated by a 0 word.

Nonexistent pages are not saved. The SAVE call also does not save the accumulators. Thus, it is possible to save all assigned nonzero memory with the one table entry 777760,,20 in AC2.

The SAVE call closes and releases the given JFN.

Can cause several software interrupts or process terminations on certain file conditions.

Generates an illegal instruction interrupt on error conditions below.

SAVE ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

SAVX1: illegal to save files on this device

IOX11: quota exceeded or disk full

All file errors can also occur.

SCTTY JSYS 324

Redefines the controlling terminal for the specified process and all of its inferiors. The controlling terminal can be redefined at any level in the job's process structure; inferior processes below this level will use this terminal by default as their controlling terminal. Therefore, the controlling terminal of a process is defined to be:

1. The one that has been explicitly defined for it via a SCTTY call.

2. If no terminal has been explicitly defined for the process, the terminal that has been explicitly defined for its closest superior via a SCTTY call.

3. If no SCTTY call has been executed for a superior process, the job's controlling terminal.

The effect of terminal interrupts on a process is dictated by the controlling terminal for the process. This means that processes that have enabled specific terminal characters will receive an interrupt when those characters are typed on the controlling terminal. If no SCTTY call has been executed for any process in the job, the controlling terminal for all processes within the job is the job's controlling terminal. (The job's controlling terminal is usually the one used to log in and control the job.) In addition to being the source of all terminal interrupts, the job's controlling terminal serves as the primary I/O designators (refer to Section 1.1) for all processes in the job, unless these designators have been changed for a process.

When a SCTTY call is executed for a process within a job, the controlling terminal and the source of terminal interrupts are changed for that process and all of its inferiors. This group of processes will receive interrupts only from the new controlling terminal and no longer from the job's controlling terminal. These processes cannot receive or change terminal interrupts from any other controlling terminals. However, primary I/O will continue to be received from and sent to the job's controlling terminal if the primary I/O designators have not been changed. For most applications, the primary I/O designators should be changed with the SPJFN call to correspond to the new controlling terminal.

ACCEPTS IN AC1: function code in the left half, and process handle in the right half

AC2: terminal designator

RETURNS +1: always

The available functions are as follows:

Code	Symbol	Meaning
0	.SCRET	Return the designator of the given process' controlling terminal. The designator is returned in AC2.
1	.SCSET	Change the given process' controlling terminal to the terminal designated in AC2. The terminal designator cannot refer to the job's controlling terminal. This function

		also changes the controlling terminal of all processes inferior to the given process.
2	.SCRST	Reset the given process' controlling terminal to the job's controlling terminal. This function also resets the controlling terminal of all processes inferior to the given process.

Functions .SCSET and .SCRST require the process to have the SC%SCT capability (refer to Section 2.6.1) enabled in its capability word.

The SCTTY monitor call cannot be used to change the controlling terminal for the current process or for any process superior to the current process.

Generates an illegal instruction interrupt on error conditions below.

SCTTY ERROR MNEMONICS:

SCTX1: invalid function code

SCTX2: terminal already in use as controlling terminal

SCTX3: illegal to redefine the job's controlling terminal

SCTX4: SC%SCT capability required

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

DESX1: invalid source/destination designator

DEVX2: device already assigned to another job

SCVEC JSYS 301

Sets the entry vector and the UUO locations for the compatibility package.

ACCEPTS IN AC1: process handle

AC2: entry vector length in the left half, and entry vector address in the right half

AC3: UUO location in the left half, and PC location in the right half

RETURNS +1: always

The compatibility package's entry vector is as follows:

Word	Symbol	Meaning
0	.SVEAD	Entry address for interpreting UUO's
1	.SVINE	Initial entry for setup and first UUO

2	.SVGET	Entry for GET share file routine (obsolete)
3	.SV40	Address to receive contents of location 40 on the UUO call
4	.SVRPC	Address to receive the return PC word on the UUO call
5	.SVMAK	Entry for MAKE share file routine (obsolete)
6 and 7	.SVCST	Communication for handling CTRL/C, START sequences between the compatibility package and the TOPS-20 Command Language

The monitor transfers to the address specified in the right half of AC2 on any monitor call whose operation code is 040-077 (i.e., monitor UUO). This transfer occurs after the monitor stores the contents of location 40 and the return PC in the locations specified by the left half and right half of AC3, respectively. The entry vector is retained but is not used by the monitor.

If AC2 is 0, the next UUO causes the compatibility package to be merged into the caller's address space. In this case, the UUO and PC locations are set from words 3 and 4, respectively, of the compatibility package's entry vector.

If AC2 is -1, UUO simulation is disabled, and an occurrence of a UUO is considered an illegal instruction. This action is useful when the user is removing UUO's from a program.

The GCVEC monitor call can be used to obtain the entry vector for the compatibility package.

SCVEC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate superior process

FRKHX3: invalid use of multiple process handle

FRKHX4: process is running

SDSTS JSYS 146

Sets the status of a device. (Refer to Section 2.4 for the descriptions of the status bits.) This call requires that the device be opened.

ACCEPTS IN AC1: JFN

AC2: mask indicating status bits to be changed

RETURNS +1: always

The SDSTS call is a no-op for devices that do not have device-dependent status bits.

The GDSTS monitor call can be used to obtain the status bits for a particular device.

Generates an illegal instruction interrupt on error conditions below.

SDSTS ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

SDVEC JSYS 543

Sets the entry vector for the Record Management System (RMS).

ACCEPTS IN AC1: process handle

AC2: entry vector length in the left half, and entry vector address in the right half

RETURNS +1: always

The Record Management System's entry vector is as follows:

Word	Symbol	Meaning
0	.SDEAD	Entry address for the RMS calls
1	.SDINE	Inital entry for the first RMS call
2	.SDVER	Pointer to RMS version block
3	.SDDMS	Address in which to store the RMS call
4	.SDRPC	Address in which to store return PC word

The GDVEC monitor call can be used to obtain the entry vector for RMS.

Generates an illegal instruction interrupt on error conditions below.

SDVEC ERROR MNEMONICS:

ILINS5: RMS facility is not available

SETER JSYS 336

Sets the most recent error condition encountered by a process. This error condition is stored in the process' Process Storage Block.

CEPTS IN AC1: process handle

AC2: error code that is to be set

RETURNS +1: always

The GETER monitor call can be used to obtain the most recent error condition encountered by a process.

Generates an illegal instruction interrupt on error conditions below.

SETER ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: process is running

SETJB JSYS 541

Sets job parameters for the specified job.

ACCEPTS IN AC1: job number, or -1 for the current job

AC2: function code

AC3: value for function

RETURNS +1: always

The available functions, along with the legal values for these functions, are described below.

Function	Values	Meaning
.SJDEN(0)		Set default for magnetic tape density.
	.SJDDN(0)	System default density
	.SJDN2(1)	200 bits/inch (8.1 rows/mm)
	.SJDN5(2)	556 bits/inch (22.5 rows/mm)
	.SJDN8(3)	800 bits/inch (32.2 rows/mm)
	.SJD16(4)	1600 bits/inch (65.3 rows/mm)
.SJPAR(1)		Set default for magnetic tape parity.
	.SJPRO(0)	Odd parity
	.SJPRE(1)	Even parity

.SJDM(2)		Set default for magnetic tape data mode.
	.SJDDM(0)	System default data mode
	.SJDMC(1)	Dump mode
	.SJDM6(2)	SIXBIT byte mode (7-track drives)
	.SJDMA(3)	ANSI ASCII mode (7 bits in 8-bit bytes)
	.SJDM8(4)	Industry compatible mode
.SJRS(3)		Set default for magnetic tape record size.
.SJDFS(4)		Set spooling mode.
	.SJSPI(0)	Immediate mode spooling
	.SJSPD(1)	Deferred mode spooling
.SJSRM(5)		Set remark for current job session. AC3 contains a pointer to the session remark, which is updated on a successful return. The first 39 characters of the session remark are placed in the job's Job Storage Block.

The SETJB monitor call requires the process to have WHEEL or OPERATOR capability enabled to set parameters for a job other than the current job.

The GETJI monitor call can be used to obtain the job parameters for a specified job.

Generates an illegal instruction interrupt on error conditions below.

SETJB ERROR MNEMONICS:

SJBX1: invalid function

SJBX2: invalid magnetic tape density

SJBX3: invalid magnetic tape data mode

SJBX4: invalid job number

SJBX5: job is not logged in

SJBX6: WHEEL or OPERATOR capability required

SJBX7: remark exceeds 39 characters

SJBX8: illegal to perform this function

SETNM JSYS 210

Sets the private name of the program being used by the current job. This name is the one printed on SYSTAT listings.

ACCEPTS IN AC1: SIXBIT name used to identify program

RETURNS +1: always

The GETNM monitor call can be used to obtain the name of the program currently being used.

SETSN JSYS 506

Sets either the system name or the private name of the program being used by the current job.

ACCEPTS IN AC1: SIXBIT name to be used as the system name. This name is the one used for system statistics.

AC2: SIXBIT name to be used as the private name. This name is the same as the one set with the SETNM call.

RETURNS +1: failure. (Currently, there are no failure returns defined.)

+2: success

System program usage statistics are accumulated in the system tables SNAMES, STIMES, and SPFLTS. (Refer to Section 2.3.2.) To make this possible, the SETSN call must be executed by each job whenever the system program name is changed. In the usual case, the TOPS-20 Command Language handles this. The argument to SETSN should be: for system programs (programs from directory <SUBSYS>), the filename, truncated to six characters and converted to SIXBIT; for private programs, "(PRIV)."

SEVEC JSYS 204

Sets the entry vector of the specified process. (Refer to Section 2.7.3.)

ACCEPTS IN AC1: process handle

AC2: entry vector word (length in the left half and address of first word in the right half), or 0

RETURNS +1: always

The GEVEC monitor call can be used to obtain the process' entry vector.

Generates an illegal instruction interrupt on error conditions below.

SEVEC ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate superior process

FRKHX3: invalid use of multiple process handle

SEVEX1: entry vector is not less than 1000

SFACS JSYS 160

Sets the ACs of the specified process.

ACCEPTS IN AC1: process handle

AC2: address of the beginning of a 20(octal) word table in the caller's address space. This table contains the values to be placed into the ACs of the specified process.

RETURNS +1: always

The specified process must not be running.

The RFACS call can be used to obtain the ACs for a specified process.

Generates an illegal instruction interrupt on error conditions below.

SFACS ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

FRKHX4: process is running

SFBSZ JSYS 46

Resets the byte size for a specific opening of a file. (Refer to the OPENF and RFBSZ calls descriptions.)

ACCEPTS IN AC1: JFN

AC2: byte size, right-justified

RETURNS +1: failure, error code in AC1

+2: success

The SFBSZ monitor call recomputes the EOF limit and the file's pointer based on the new byte size given.

SFBSZ ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX5: file is not open

DESX8: file is not on disk

SFBSX1: illegal to change byte size for this opening of file

SFBX2: invalid byte size

SFCOC JSYS 113

Sets the control character output control (CCOC) for the specified terminal. (Refer to Section 2.4.3.2 and the RFCOC call description.)

ACCEPTS IN AC1: file designator

AC2: control character output control word

AC3: control character output control word

RETURNS +1: always

The CCOC words consist of 2-bit bytes, each byte representing the output control for one of the ASCII codes 0-37.

The SFCOC call is a no-op if the designator is not associated with a terminal.

The RFCOC monitor call can be used to obtain the CCOC words for a specified terminal.

SFCOC ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

SFMOD JSYS 110

Sets the program-related modes for the specified terminal. The modes that can be set by this call are in the following bits of the JFN mode word. (Refer to Section 2.4.3.1.)

B0(TT%OSP)	output suppression control
B18-B23(TT%WAK)	wakeup control
B24(TT%ECO)	echoes on
B28-B29(TT%DAM)	data mode

ACCEPTS IN AC1: file designator

AC2: JFN mode word

RETURNS +1: always

The SFMOD call is a no-op if the designator is not associated with a terminal.

The STPAR monitor call can be used to set device-related modes of the JFN mode word, and the RFMOD monitor call can be used to obtain the JFN mode word.

SFMOD ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

SFORK JSYS 157

Starts the specified process. If the process is frozen, the SFORK call changes the PC but does not resume the process. The RFORK call must be used to resume the process.

ACCEPTS IN AC1: process handle

AC2: address of word containing the PC for starting the process. This address is in the current process.

RETURNS +1: always

The SFRKV monitor call can be used to start a process at a given position in its entry vector.

Generates an illegal instruction interrupt on error conditions below.

SFORK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

SFPOS JSYS 526

Sets the position of the specified terminal's pointer. (Refer to Section 2.4.3.4 for information on page lengths and widths of terminals.)

ACCEPTS IN AC1: file designator

AC2: position within a page (i.e., line number) in the left half, and position with a line (i.e., column number) in the right half

RETURNS +1: always

The SFPOS monitor call is a no-op if the designator is not associated with a terminal or is in any way illegal.

The RFPOS monitor call can be used to obtain the current position of the terminal's pointer.

SFPOS ERROR MNEMONICS:

TTYX01: line is not active

SFPTR JSYS 27

Sets the position of the specified file's pointer for subsequent I/O to the file. The SFPTR call specifying a certain byte number, followed by a BIN call, has the same effect as a RIN call specifying the same byte number.

ACCEPTS IN AC1: JFN

AC2: byte number to which the pointer is to be set, or -1 to set the pointer to the current end of the file

RETURNS +1: failure, error code in AC1

+2: success

The RFPTR monitor call can be used to obtain the current position of the file's pointer.

SFPTR ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX8: file is not on disk

SFPTX1: file is not open

SFPTX2: illegal to reset pointer for this file

SFPTX3: invalid byte number

SFRKV JSYS 201

Starts the specified process using the given position in its entry vector.

ACCEPTS IN AC1: process handle

AC2: position (0-n) in the entry vector to use for the start address. Position 0 is always the primary start address, and position 1 is the reenter address.

RETURNS +1: always

The process is started at the specified position in the entry vector, not at the location pointed to by the entry vector word.

If the process has a TOPS-10 format entry vector (JRST in the left half) (for example, it was obtained from a TOPS-10 format save file via a GET call), the entry vector position 0 means "use the contents of .JBSA=120 as the start address," and position 1 means "use the contents of .JBREN=124 as the reenter address."

Generates an illegal instruction interrupt on error conditions below.

SFRKV ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

FRKHX4: process is running

SFRVX1: invalid position in entry vector

SFTAD JSYS 534

Sets the dates and times associated with the specified file.

ACCEPTS IN	AC1:	source designator
	AC2:	address of argument block
	AC3:	length of argument block
RETURNS	+1:	always

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.RSWRT	Internal date and time file was last written.
1	.RSCRV	Internal date and time file was created.
2	.RSREF	Internal date and time file was last referenced.
3	.RSCRE	Internal system date and time of last write.

The values in the argument block for the number of words (i.e., length) given in AC3 are set for the file. These values are checked against the current date and time. Values greater than the current date and time can be set only if the process has WHEEL or OPERATOR capability enabled. In addition, the process must have WHEEL or OPERATOR capability enabled to set the internal system date (.RSCRE).

If the designator represents a device for which dates are meaningless (e.g., dates for terminals), or if any value given is -1, the given value is ignored, and the current date, if pertinent, is not changed. If the argument block has more than four words, given values for these words are checked to be in valid format and then ignored, if valid.

To set dates for disk files, the process must have write or owner access to the file.

The RFTAD monitor call can be used to obtain the dates and times associated with a specified file.

Generates an illegal instruction interrupt on error conditions below.

SFTAD ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX3:	JFN is not assigned
DESX7:	JFN cannot refer to output wildcard designators
DATE6:	system date and time not set
STADX2:	invalid date or time
?x<4hv2:	illegal to change specified bits
OPNX25:	device is write locked
CAPX1:	WHEEL or OPERATOR capability required

SFUST JSYS 551

Sets the name of either the author of the file or the user who last wrote to the file. This monitor call requires the process to have WHEEL or OPERATOR capability enabled to set the writer's name or to have write or owner access to the file to set the author's name.

ACCEPTS IN AC1: function code in the left half, and JFN of the file in the right half

AC2: pointer to ASCIZ string containing the name

RETURNS +1: always, with an updated string pointer in AC2

The defined functions are as follows:

Code	Symbol	Meaning
0	.SFAUT	Set the name of the author of the file.
1	.SFLWR	Set the name of the user who last wrote the file.

The GFUST monitor call can be used to return the name of either the author of the file or the user who last wrote the file.

Generates an illegal instruction interrupt on error conditions below.

SFUST ERROR MNEMONICS:

SFUSX1:	invalid function
SFUSX2:	insufficient system resources
SFUSX4:	file expunged
SFUSX5:	write or owner access required
SFUSX6:	no such user name
DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX7:	JFN cannot refer to output wildcard designators
DESX8:	file is not on disk
DESX10:	structure is dismounted
CAPX1:	WHEEL or OPERATOR capability required

SIBE JSYS 102

Tests to see if the designated file input buffer is empty.

ACCEPTS IN AC1: source designator

RETURNS +1: input buffer is not empty. Number of bytes remaining in input buffer is returned in AC2 if designator refers either to a JFN opened for read access or to an active terminal.

+2: input buffer is empty if designator refers either to a JFN opened for read access or to an active terminal. This return is also taken if the designator refers either to a JFN not opened for read access (AC2 contains 0) or to an inactive terminal (AC2 contains the appropriate error code).

The SOBE monitor call can be used to determine if the output buffer is empty, and the SOBF monitor call can be used to determine if the output buffer is full.

SIBE ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

SIN JSYS 52

Reads a string from the specified source into the caller's address space. The string can be a specified number of bytes or terminated with a specified byte.

ACCEPTS IN AC1: source designator

AC2: pointer to string in the caller's address space

AC3: count of number of bytes in string, or 0

AC4: byte (right-justified) on which to terminate input (optional)

RETURNS +1: always, updated string pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 controls the number of bytes to read.

AC3=0	The string being read is terminated with a 0 byte.
AC3>0	A string of the specified number of bytes is to be read or a string terminated with the byte given in AC4 is to be read, whichever occurs first.
AC3<0	A string of minus the specified number of bytes is to be read.

The contents of AC4 is ignored unless the contents of AC3 is a positive number.

The input is terminated when the byte count becomes 0, the specified terminating byte is reached, the end of the file is reached, or an error occurs during the transfer. The program can process an end-of-file condition if an ERJMP or ERCAL is the next instruction following the SIN call.

After execution of the call, the file's pointer is updated for subsequent I/O to the file. AC2 is updated to point to the last byte read or, if AC3 contained 0, the last nonzero byte read. The count in AC3 is updated toward zero by subtracting the number of bytes read from the number of bytes requested to be read. If the input was terminated by an end-of-file interrupt, AC1 through AC3 are updated (where pertinent) to reflect the number of bytes transferred before the end of the file was reached.

When the SIN call is used to read data from a magnetic tape, the size of the records to read is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) words. The record size must be at least as large as the largest record being read from the tape. The SIN call will read across record boundaries on the tape until it reads the number of bytes requested by the contents of AC3. The call gives the data to the program with no indication of tape marks. Thus, if the record is 1000 bytes and a SIN call is given requesting 2000 bytes, it would return two full records to the program.

When reading in reverse, the number of bytes requested (i.e., the count in AC3) and the record size should equal the actual size of the record on the tape. (Refer to Section 2.4.2.1 for more information about magnetic tape I/O.)

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

SIN ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX5:	file is not open
IOX1:	file is not open for reading

IOX4: end of file reached

IOX5: device or data error

IOX7: insufficient system resources (Job Storage Block full)

IOX8: monitor internal error

SINR JSYS 531

Reads a record from the specified magnetic tape into the caller's address space. The maximum size of the record to read is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) words. Refer to Section 2.4.2.1 for more information about magnetic tape I/O.

ACCEPTS IN AC1: source designator

AC2: pointer to string in the caller's address space

AC3: count of number of bytes in string, or 0

AC4: byte (right-justified) on which to terminate input (optional)

RETURNS +1: always, updated string pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 and AC4 are interpreted in the same manner as they are in the SIN monitor call.

Each SINR call returns one record to the caller. Thus, the caller can read variable-length records by indicating in AC3 the number of bytes to read. Upon execution of the call, AC3 is updated to reflect the number of bytes read (i.e., the number of bytes in the record).

The number of bytes read depends on the number of bytes requested and the record size. When using SINR, the program must set the record size to a value greater than or equal to the actual size of the largest record being read from the tape, or an error (IOX5) will be returned. If the SINR call requests the same number of bytes as the record size, the requested number is given to the caller. When the record size equals the size of the actual record, all bytes in the record are read, and AC3 contains 0 on return. When the record size is larger than the actual record, all bytes of the record are read, but AC3 contains the difference of the number requested and the number read. If the SINR call requests fewer bytes than in the actual record, the requested number is given to the caller, the remaining bytes are discarded, and an error (IOX10) is returned. In all cases, the next request for input begins reading at the first byte of the next record on the tape because a SINR call never reads across record boundaries.

When reading in reverse, the number of bytes requested (i.e., the count in AC3) should be at least as large as the size of the record on the tape. If the requested number is smaller, the remaining bytes in the record are discarded from the beginning of the record.

The action taken on a SINR call differs from the action taken on a SIN call. The SIN call reads across record boundaries to read all the bytes in a file. The SINR call does not read across record boundaries and will discard some bytes in the file if the requested number is smaller than the actual record.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

SINR ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

IOX1: file is not open for reading

IOX4: end of file reached

IOX5: device or data error

IOX7: insufficient system resources (Job Storage Block full)

IOX8: monitor internal error

IOX10: record is longer than user requested

SIR JSYS 125

Sets the channel and priority level table addresses for the specified process. (Refer to Section 2.5.3.) These addresses are in the specified process' address space.

ACCEPTS IN AC1: process handle

AC2: address of the priority level table in the left half, and address of the channel table in the right half

RETURNS +1: always. The addresses in AC2 are stored in the Process Storage Block.

If the contents of the tables are changed after execution of the SIR call, the new contents will be used on the next interrupt.

The RIR monitor call can be used to obtain the table addresses for a specified process.

Generates an illegal instruction interrupt on error conditions below.

SIR ERROR MNEMONICS:

SIRX1: table address is not greater than 20

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

SIRCM JSYS 142

Sets the mask for reserved software interrupt channels for the specified inferior process. Conditions occurring on software channels that have the corresponding mask bit set do not generate an interrupt to the inferior process. Instead, the conditions cause the process to terminate or freeze.

ACCEPTS IN AC1: inferior process handle

AC2: channel mask with bits set for reserved channels

AC3: deferred terminal interrupt word

RETURNS +1: always

The RIRCM monitor call can be used to obtain the mask for reserved software interrupt channels. Although a process can read its own channel mask, it cannot set its own; the SIRCM call can be given only for inferior processes. This call provides a facility for a superior process to monitor an inferior one (e.g., illegal instructions, memory traps). However, if the inferior process contains an ERJMP or ERCAL symbol after instructions that generate an interrupt on failure, the ERJMP or ERCAL will prevent the generation of the interrupt. Thus, the superior will not be able to monitor the inferior with the SIRCM call.

Generates an illegal instruction interrupt on error conditions below.

SIRCM ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

SIZEF JSYS 36

Returns the length of an existing file.

ACCEPTS IN AC1: JFN

RETURNS +1: failure, error code in AC1

+2: success, byte count that referenced the last byte written into the file in AC2, and number of pages (512 words) in file in AC3. The byte count returned depends on the byte size recorded in the FDB and not on the byte size specified in the OPENF call.

For a file with holes, the byte count in AC2 does not reflect the file's actual size.

The GTFDB monitor call can be used to obtain the byte size in which the file was written.

SIZEF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

SJPRI JSYS 245

Sets the job priority by specifying the scheduler priority control word for any job. The priority word is set in the top process of the designated job, but not in any existing inferior processes. However, it is passed down when a new inferior process is created. This call requires the process to have WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: job number

AC2: priority word

RETURNS +1: always

The priority word contains a percentage of CPU resources in the left half and 0 in the right half. By placing the desired percentage (1-99) in the left half, a job can be guaranteed a certain percentage of CPU time. A priority word of 0 indicates no special priority.

Generates an illegal instruction interrupt on error conditions below.

SJPRI ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

SJPRX1: job is not logged in

SKPIR JSYS 127

Tests to see if the software interrupt system is enabled for the specified process.

ACCEPTS IN AC1: process handle

RETURNS +1: failure, software interrupt system is off

+2: success, software interrupt system is on

The EIR monitor call is used to enable the software interrupt system, and the DIR monitor call is used to disable the system.

Generates an illegal instruction interrupt on error conditions below.

SKPIR ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

SMON JSYS 6

Sets various monitor flags. The SMON monitor call requires the process to have WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: function code

AC2: new value for the indicated function

RETURNS +1: always

The codes for the functions are as follows:

Code	Symbol	Meaning
0	.SFFAC	FACT file entries are allowed.
1	.SFCDE	CHECKD found errors.
2	.SFCDR	CHECKD is running.
3	.SFMST	Manual start is in progress.
4	.SFRMT	Remote LOGINs (dataset lines) are allowed.
5	.SFPTY	PTY LOGINs are allowed.
6	.SFCTY	CTY LOGINs are allowed.
7	.SFOPR	Operator is in attendance.
10	.SFLCL	Local LOGINs (hardwired lines) are allowed.

11	.SFBTE	Bit table errors found on startup.
12	.SFCRD	Users can change nonprivileged directory parameters with the CRDIR monitor call.
13	.SFNVT	ARPANET terminal LOGINs are allowed.
21	.SFUSG	USAGE file entries are allowed.
22	.SFFLO	Disk latency optimization using the RH20 backup register is enabled. This feature is not to be enabled unless the M8555 board of the RH20 is at Revision Level D AND either of the KL10-C processor is at Revision Level 10 or KL10-E processor is at Revision Level 2.
44	.SFNTN	Turn ARPANET on.
45	.SFNDU	Reinitialize ARPANET if it is down.
46	.SFNHI	Initialize ARPANET host table.
47	.SFTMZ	Set the local time zone to the value given in AC2.
50	.SFLHN	Set the local ARPANET host number to the value given in AC2.
51	.SFAVR	Account validation will be running on this system.

Function codes 0 through 22 represent a specific monitor flag bit. When the value of the function is 1 (i.e., AC2 contains the value 1), the bit corresponding to the function is set. When the value is 0, the bit is cleared.

The TMON monitor call can be used to obtain the settings of the various monitor flags.

Generates an illegal instruction interrupt on error conditions below.

SMON ERROR MNEMONICS:

SMONX1: WHEEL or OPERATOR capability required

SMONX2: invalid SMON function

SNOOP JSYS 516

Performs system performance analysis. The SNOOP call requires the process to have WHEEL or OPERATOR capability enabled, because the process can patch any instruction in the monitor with this call. For example, the user program can build a PC histogram by patching an instruction in the code for the 1.0-millisecond clock.

The general procedure for using the SNOOP call is as follows:

1. The user program supplies a set of breakpoint routines that are called by the monitor when control reaches one of the patched instructions. These routines are mapped into the monitor's address space into an area selected by the monitor. Thus, the routines must have self-relocating code or must be relocated by the user program to where they will be run, based on the monitor address supplied by the monitor.

2. The user program defines a number of breakpoints, analogous to DDT breakpoints.

3. The user program inserts all of the breakpoints simultaneously.

4. The user program goes to "sleep" or waits for terminal input while its breakpoint routines obtain control.

5. When the user program determines that the routines have completed, it removes the breakpoints.

The user program breakpoint routines run in the monitor address space, which means that the addresses of the code and the data are monitor addresses. The user program must modify these addresses, based on the values returned by the monitor, after the initialization but before the "snooping." The breakpoint routines must preserve any accumulators they use. Also, they must not cause a page fault if at interrupt level or if a patch has been made in the page fault handler or in the scheduler. Thus, the breakpoint routines should test for swappable code being in memory before referencing it. If swappable code needs to be referenced, the swappable monitor can be locked in memory, if desired. When a patch is made to a routine called at many interrupt levels, the program must specify a reentrant instruction to be used for patching.

ACCEPTS IN AC1: function code

AC2: arguments for

AC3: the specified

AC4: function

RETURNS +1: failure, error code in AC1

+2: success

The following functions are available:

Function Code	Symbol	Meaning
0	.SNPLC	Declare and lock code into the monitor's address space. AC2: number of pages desired AC3: page number in user space of start of breakpoint routines to be locked

On return, the pages are locked contiguously in the monitor's address space, and AC2 contains the monitor page numbers corresponding to the given user page number.

1	.SNPLS	Lock the swappable monitor. This function is useful for analyzing swappable data at interrupt level. On return, the entire swappable monitor is locked.
2	.SNPDB	Define a breakpoint

AC2: number of breakpoint

AC3: address in monitor space to be patched. The patched instruction can be a skip type instruction or a PUSHJ instruction, and the patching is similar to that in DDT. The routines will receive control before the patched instruction is executed.

AC4: instuction to be executed before the patched instruction is executed. The instruction can be:

JSR LOC where LOC is an address in monitor space of the user's routine.

PUSHJ P,LOC when reentrant or recursive code is patched.

AOS LOC to count frequency of monitor execution points.

The error return is given if breakpoints have already been inserted.

3	.SNPIB	Insert all breakpoints and start analyzing.
4	.SNPRB	Remove all breakpoints and stop analyzing.
5	.SNPUL	Unlock and release all storage, and undefine and remove all breakpoints.
6	.SNPSY	Obtain the address of a monitor symbol.

AC2: radix-50 symbol

AC3: radix-50 program name if a local address is desired. If AC3 is 0, the entire symbol table is searched.

On return, AC2 contains the monitor address or value of the symbol.

7	.SNPAD	Obtain a monitor symbol. AC2: 36-bit value of symbol that is to be looked up in the monitor's symbol table. AC3: radix-50 program name if a local value is desired. If AC3 is 0, the entire symbol table is searched. On return, AC2 contains the first radix-50 monitor symbol that is closest to and has a value less than the specified value, and AC3 contains the difference between the value of the symbol returned and the specified value.

SNOOP ERROR MNEMONICS:

SNOPX1: WHEEL or OPERATOR capability required

SNOPX2: invalid function

SNOPX3: .SNPLC function must be first

SNOPX4: only one .SNPLC function allowed

SNOPX5: invalid page number

SNOPX6: invalid number of pages to lock

SNOPX7: illegal to define breakpoints after inserting them

SNOPX8: breakpoint is not set on instruction

SNOPX9: no more breakpoints allowed

SNOPX10: breakpoints already inserted

SNOPX11: breakpoints not inserted

SNOPX12: invalid format for program name symbol

SNOPX13: no such program name symbol

SNOPX14: no such symbol

SNOPX15: not enough free pages for snooping

SNOPX16: multiply-defined symbol

SNOPX17: breakpoint already defined

SNOPX18: data page is not private or copy-or-write

SOBE JSYS 103

Tests to see if the designated file output buffer is empty.

ACCEPTS IN AC1: destination designator

RETURNS +1: output buffer is not empty. Number of bytes remaining in output buffer is returned in AC2.

+2: output buffer is empty; AC2 contains 0. This return is given if an error occurs on the call; AC2 contains the appropriate error code.

If the designator is not associated with a terminal, the +2 return is given.

The SIBE call can be used to determine if the input buffer is empty.

SOBE ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

SOBF JSYS 175

Tests to see if the designated file output buffer is full.

ACCEPTS IN AC1: file designator

RETURNS +1: output buffer is not full. This return is given if an error occurs on the call; AC2 will contain 0.

+2: output buffer is full

On either return, the number of bytes remaining in the output buffer is returned in AC2 (if no error occurred on the call).

SOBF ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DESX6: file must be a terminal

DEVX2: device already assigned to another job

TTYX01: line is not active

SOUT JSYS 53

Writes a string from the caller's address space to the specified destination. The string can be a specified number of bytes or terminated with a specified byte.

ACCEPTS IN AC1:		destination designator
	AC2:	pointer to string to be written
	AC3:	count of the number of bytes in string, or 0
	AC4:	byte (right-justified) on which to terminate output
RETURNS	+1:	always, updated string pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 controls the number of bytes to write.

AC3=0	The string being written is terminated with a 0 byte.
AC3>0	A string of the specified number of bytes is to be written or a string terminated with the byte given in AC4 is to be written, whichever occurs first.
AC3<0	A string of minus the specified number of bytes is to be written.

The contents of AC4 is ignored unless the contents of AC3 is a positive number.

The output is terminated when the byte count becomes 0, the specified terminating byte is reached, or an error occurs during the transfer. The specified terminating byte is copied to the destination.

After execution of the call, the file's pointer is updated for subsequent I/O to the file. AC2 is updated to point to the last byte written or, if AC3 contained 0, the last nonzero byte written. The count in AC3 is updated toward zero by subtracting the number of bytes written from the number of bytes requested to be written.

When the SOUT call is used to write data to a magnetic tape, it sends a series of bytes packed into records of the specified record size. The size of the records to write is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) words. Thus, if the record size is 1000 bytes, two SOUT calls, each writing 500 bytes, would write one record. If during the writing, the end of tape mark was passed, an error (IOX5) is given. However, the data has been successfully written and the device status word has the MT%EOT bit set to indicate this condition. Refer to Section 2.4.2.1 for more information about magnetic tape I/O.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

SOUT ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX5:	file is not open
IOX2:	file is not opened for writing
IOX5:	device or data error
IOX6:	illegal to write beyond absolute end of file
IOX7:	insufficient system resources (Job Storage Block full)
IOX8:	monitor internal error
IOX11:	quota exceeded or disk full

SOUTR JSYS 532

Writes a variable-length record from the caller's address space to the specified magnetic tape. The maximum size of the record to write is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) words. (Refer to Section 2.4.2.1 for more information about magnetic tape I/O.)

ACCEPTS IN	AC1:	destination designator
	AC2:	pointer to string to be written
	AC3:	count of number of bytes in string, or 0
	AC4:	byte (right-justified) on which to terminate output (optional)
RETURNS	+1:	always, updated string pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 and AC4 are interpreted in the same manner as they are in the SOUT monitor call.

Each SOUTR call writes at least one record. Thus, the caller can write variable-length records by indicating in AC3 the number of bytes to write in the record. If the SOUTR call requests more bytes to be written than the maximum record size, then records of the maximum size are written, plus another record containing the remaining bytes. If the SOUTR call requests fewer bytes than the maximum, or a number equal to the maximum, to be written, then records of the requested size are written.

The SOUTR call differs from the SOUT call in that the SOUTR call writes records on the tape upon execution of the call. The SOUT call does not write a record on the tape until the number of bytes equal to the record size have been written. Thus, if a record is being made from several strings in the caller's address space, the SOUT call can be used for the first strings and the SOUTR call for the last string.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

SOUTR ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX3:	JFN is not assigned
DESX5:	file is not open
IOX2:	file is not open for writing
IOX6:	illegal to write beyond absolute end of file
IOX7:	insufficient system resources (Job Storage Block full)
IOX8:	monitor internal error
IOX9:	function legal for sequential write only
IOX11:	quota exceeded or disk full

SPACS JSYS 60

Sets the accessibility of a page. This call affects the map word directly indicated by the argument (i.e., no indirect pointers are allowed).

ACCEPTS IN AC1: process/file designator in the left half, and page number within the file in the right half

AC2: access information

B2(PA%RD)	permit read access
B3(PA%WT)	permit write access
B4(PA%EX)	permit execute access
B9(PA%CPY)	copy-on-write

RETURNS +1: always

When used to modify a process page, the SPACS call does not allow any greater access than can be obtained with the PMAP call (i.e., the access specified on the OPENF call is applied to SPACS operations involving file pointers).

The SPACS call does not allow bits to be set in a page that does not already exist.

The RPACS monitor call can be used to obtain the accessibility of a page.

Generates an illegal instruction interrupt on error conditions below.

SPACS ERROR MNEMONICS:

DESX1:	invalid source/destination designator
DESX2:	terminal is not available to this job
DESX3:	JFN is not assigned
DESX4:	invalid use of terminal designator or string pointer
DESX5:	file is not open
DESX8:	file is not on disk
SPACX1:	invalid access requested
FRKHX1:	invalid process handle
FRKHX2:	illegal to manipulate a superior process
FRKHX3:	invalid use of multiple process handle

SPJFN JSYS 207

Sets the primary JFNs of the specified process.

ACCEPTS IN AC1: process handle

AC2: primary input JFN in the left half, and primary output JFN in the right half

RETURNS +1: always

The JFNs given cannot be either 100 or 101. These JFNs cause the specified process to receive an error on any primary I/O operation.

The GPJFN monitor call can be used to obtain the primary JFNs.

Generates an illegal instruction interrupt on error conditions below.

SPJFN ERROR MNEMONICS:

FRKHX1:	invalid process handle
FRKHX2:	illegal to manipulate a superior process
FRKHX3:	invalid use of multiple process handle
DESX3:	JFN is not assigned

SPLFK JSYS 314

Splices a process structure. The process that becomes the new superior must be either the one executing the SPLFK monitor call or an inferior of it. The new inferior process must be an inferior of the executing process. The new superior process must not be the same process as the new inferior process and must not be inferior to the new inferior process. The new inferior process and all of its inferiors will be frozen after execution of the SPLFK call.

ACCEPTS IN AC1: process handle of the new superior process

AC2: process handle of the new inferior process

RETURNS +1: failure, error code in AC1

+2: success, a process handle in AC1. This handle may be used by the new superior process (in AC1) to refer to its new inferior (in AC2).

SPLFK ERROR MNEMONICS:

SPLFX1: process is not inferior or equal to self

SPLFX2: process is not inferior to self

SPLFX3: new superior process is inferior to intended inferior

FRKHX1: invalid process handle

SPOOL JSYS 517

Defines and initializes a device to be used for input spooling or sets and reads the directory for a spooled device.

ACCEPTS IN AC1: length of argument block in the left half, and function code in the right half

AC2: address of argument block

RETURNS +1: failure, error code in AC1

+2: success

The format of the argument block is different depending upon the particular function desired. The available functions, along with their argument block formats, are as follows:

<table>
<tr><th>Code</th><th>Symbol</th><th>Meaning</th></tr>
<tr><td>0</td><td>.SPLDI</td><td>Define an input spooling device. The argument block is:
<table>
<tr><th>Word</th><th>Symbol</th><th>Meaning</th></tr>
<tr><td>0</td><td>.SPLDV</td><td>Device designator of input device.</td></tr>
<tr><td>1</td><td>.SPLNA</td><td>Pointer to name string comprising the set of files to be input.</td></tr>
<tr><td>2</td><td>.SPLGN</td><td>Generation number of first file. This number is incremented by 1 each time the spooled device is opened.</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>.SPLSD</td><td>Set the directory of the spooled device. The argument block is:
<table>
<tr><th>Word</th><th>Symbol</th><th>Meaning</th></tr>
<tr><td>0</td><td>.SPLDV</td><td>Device designator of spooled device.</td></tr>
<tr><td>1</td><td>.SPLDR</td><td>Directory number. This number is the logged-in directory number of the user who opened the spooled device.</td></tr>
</table>
This function requires the process to have WHEEL or OPERATOR capability enabled.</td></tr>
<tr><td>2</td><td>.SPLRD</td><td>Read the directory of the spooled device. The argument block is:
<table>
<tr><th>Word</th><th>Symbol</th><th>Meaning</th></tr>
<tr><td>0</td><td>.SPLDV</td><td>Designator of spooled device.</td></tr>
</table>
The directory number of the spooled device is returned in word 1 of the argument block.</td></tr>
</table>

To read from a spooled input device, the user first defines the name of the files comprising his set of spooled input files. The files have names in the format:

```
STR:<SPOOLED-DIRECTORY>DEVICE-DIR#.NAME.1,2,3,...
```

The spooled directory is the directory to receive any spooled input from the device. The .SPLSD function can be used by a privileged process to set the directory. The default directory for all of the spooled devices is <SPOOL>.

The device is the name of the device being used for spooled input. It is the same name that was given on the original GTJFN call.

The directory number is the logged-in directory number of the user that opened the spooled device.

The name is the name of the set of files to be input. The .SPLDI function is used to define this name.

The generation number begins with the value specified by the .SPLDI function and increments by one each time the spooled device is opened.

Thus, if the input spooler for the card reader (CDR) is reading files for a user whose directory number is 23, then the files might have names like

<SPOOL>CDR-23.BATCH-SEQUENCE-37.1,2,3,...

To initialize the spooled card reader, the user would then execute the SPOOL call giving "BATCH-SEQUENCE-37" as the name of the set of files to be input and "1" as the beginning generation number.

SPOOL ERROR MNEMONICS:

SPLX1: invalid function

SPLX2: argument block too small

SPLX3: invalid device designator

SPLX4: WHEEL or OPERATOR capability required

SPLX5: illegal to specify 0 as generation number for first file

SPLX6: no directory to write spooled files into

SPRIW JSYS 243

Sets the priority word for the specified process. This call requires the process to have WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: process handle

AC2: priority word

RETURNS +1: always

Refer to the SJPRI monitor call description for the format of the priority word.

Generates an illegal instruction interrupt on error conditions below.

SPRIW ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

SSAVE JSYS 203

Creates a sharable save format file for the given JFN by copying (not sharing) pages from the given process. (Refer to Section 2.7.2 for the format of a sharable save file.) This monitor call is used for creating shared programs. It saves the file in groups of contiguous pages for which the same access is desired.

ACCEPTS IN	AC1:	process handle in the left half, and JFN in the right half
	AC2:	one table entry, or 0 in the left half and the address of the table in the right half (see below)
	AC3:	flag bits (not yet implemented; AC3 should be 0)
RETURNS	+1:	always

The table has a one-word entry for each group of pages and is terminated by a zero word. Each word has the following format:

Bit	Symbol	Meaning
0-17	SS%NNP	Negative of the number of pages in each group (right-justified).
18	SS%CPY	Allow copy-on-write access to the group of pages.
19	SS%UCA	Limit the access according to the current access of the user's page. (See below.)
20	SS%RD	Allow read access to the group of pages.
21	SS%WR	Allow write access to the group of pages.
22	SS%EXE	Allow execute access to the group of pages.
27-35	SS%FPN	Number of the first page in the group (right-justified).

When B19(SS%UCA) is set, the access to the group of pages is determined by ANDing the access bits specified in the table word with the corresponding access bits for the user's pages (as determined by the RPACS call). This means that a given access is allowed only if both the SSAVE call indicates it and the page currently has it. If B19(SS%UCA) is not set, the access granted to the group of pages is that indicated by the bits set in the table word.

The SSAVE call does not save the accumulators nor does it save nonexistent pages.

The GET monitor call is used to map a file saved with the SSAVE call back into a given process.

Can cause several software interrupts or process terminations on certain file conditions.

Generates an illegal instruction interrupt on error conditions below.

SSAVE ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

SSAVX1: illegal to save files on this device

SSAVX2: page count is not less than or equal to 1000

SSAVX3: insufficient system resources (Job Storage Block full)

SSAVX4: directory area of EXE file is more than one page

IOX11: quota exceeded or disk full

All I/O errors can also occur.

STAD JSYS 226

Sets the system's date. (Refer to Section 2.8.2.)

ACCEPTS IN AC1: day in the left half, and fraction of the day in the right half

RETURNS +1: failure, error code in AC1

+2: success

The STAD call requires the process to have WHEEL or OPERATOR capability enabled if the system's date is already set.

The GTAD monitor call can be used to obtain the system's date.

STAD ERROR MNEMONICS:

STADX1: WHEEL or OPERATOR capability required

STADX2: invalid date or time

STCMP JSYS 540

Compares two ASCIZ strings in the caller's address space. Note that letters are always considered as upper case, regardless of their case within the string. Therefore, the strings ABC and abc are considered an exact match.

ACCEPTS IN AC1: pointer to test string

AC2: pointer to base string

RETURNS +1: always, with

AC1 containing the compare code:

B0(SC%LSS) Test string is less than base string.

B1(SC%SUB) Test string is a subset of base string.

B2(SC%GTR) Test string is greater than base string.

AC2 containing base string pointer, updated such that an ILDB instruction will reference the first nonmatching byte.

One string is considered less than another string if the ASCII value of the first nonmatching character in the first string is less than the ASCII value of the character in the same position in the second string.

One string is considered a subset of another string if both of the following conditions are true:

1. From left to right, the ASCII values of the characters in corresponding positions are the same.

2. The test string is shorter than the base string.

Two strings are considered equal if the ASCII values of the characters in corresponding positions are the same and the two strings are the same size. In this case, the contents of AC1 is 0 on return.

STDEV JSYS 120

Translates the given string to its corresponding device designator.

ACCEPTS IN AC1: pointer to the string to be translated

RETURNS +1: failure, error code in AC2

+2: success, device designator (refer to Section 2.4) in AC2

The string to be translated is terminated by the first space (ASCII code 40), null (ASCII code 0), or colon (ASCII code 72).

The DEVST monitor call can be used to translate a device designator to its corresponding string.

STDEV ERROR MNEMONICS:

STDVX1: no such device

STI JSYS 114

Simulates terminal input.

ACCEPTS IN AC1: file designator (only terminal designators are legal)

AC2: character to be input, right-justified

RETURNS +1: always

The character is taken from the accumulator and placed into the specified terminal's input buffer whether or not the buffer is empty. The DIBE call can be used to prevent sending an interrupt character (e.g., CTRL/C) before the program has processed all of the previous input.

The STI monitor call requires the process to have WHEEL or OPERATOR capability enabled if the specified terminal either is not assigned or opened by the process or is not accepting advice. (Refer to the TLINK bit TT%AAD.)

The use of this monitor call is not recommended for pseudo-terminals (PTYs). The recommended procedure for placing a character in the PTY input buffer is to open the PTY for output with OPENF and then perform output with the BOUT call.

Generates an illegal instruction interrupt on error conditions below.

STI ERROR MNEMONICS:

TTYX1: device is not a terminal

DESX2: terminal is not available to this job

DEVX2: device already assigned to another job

WHELX1: WHEEL or OPERATOR capability required

TTYX01: line is not active

STIW JSYS 174

Sets the terminal interrupt word (refer to Section 2.5.6) for the entire job or a specific process. This call declares that terminal characters that usually cause an interrupt are instead to be passed to the program as input. In actuality, the STIW call sets the interrupt word mask, thus determining for each of the 36 terminal codes if the job or process should receive an interrupt. The call's effect is different, depending on whether the call is being executed for the entire job or for a specific process in the job.

When the STIW call is executed for the entire job, codes corresponding to the bits on in the mask will cause an interrupt if a process in the job has enabled for an interrupt on that code. If multiple processes have enable that code, the lowest inferior process receives the interrupt. (If several processes at the same lowest level have enabled the code, the process that receives the interrupt is determined by the system.) If no process has enabled that code, the character corresponding to the code is passed to the program. Also, characters are passed to the program when their corresponding bits are off in the mask, even if a process has enabled that code. Initially, all codes are declared to cause an interrupt (i.e., all bits in the mask are on), and the program can execute the RTIW call to determine the current status. Thus if the program wishes to read a terminal interrupt character as input, it executes the STIW call for the entire job and turns off the mask bit corresponding to the character.

When the STIW call is executed for a specific process in the job, codes corresponding to the bits on in the mask are assumed to be enabled by the specific process and cause an interrupt if in fact they are enabled. If the process has not enabled for the code, the character corresponding to the code is ignored, if it is typed. Characters corresponding to the bits off in the mask are assumed not to be enabled by the process. This use of the STIW call is implicitly executed on an ATI call.

Each time the STIW call is executed for a specific process, the mask is changed to reflect the bits changed in that process.

The STIW call sets or clears specific terminal codes for a particular process without actually changing the channel assignment that each code has. The ATI call is used to set the channel assignment, and the DTI call is used to clear the assignment.

The STIW call requires the process to have SC%CTC capability enabled to disable the code for CTRL/C interrupts or to give -5 as an argument.

```
ACCEPTS IN AC1:  B0(ST%DIM) set the deferred terminal  interrupt  mask
                            given in AC3

                 B18-B35    process handle, or -5 for entire job
                 (ST%PRH)

            AC2: terminal interrupt word mask.
                 Bit n on means terminal code n is enabled.

            AC3: deferred terminal interrupt word mask.
                 Bit n on means terminal code n is deferred.

RETURNS      +1: always
```

The argument in AC3 is ignored, and no change is made to the deferred interrupt word mask, if B0(ST%DIM) is not set or if the process handle in AC1 does not indicate a specific process.

If multiple processes enable the same interrupt character and any one of the processes declares it deferred, the character is deferred for all the processes that enabled it.

The RTIW call can be used to obtain the terminal interrupt word masks.

STIW ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

STO JSYS 246

Simulates terminal output.

ACCEPTS IN AC1: file designator (only terminal designators are legal)

RETURNS +1: always, with the character right-justified in AC2

The character is taken from the specified terminal's output buffer and placed in the accumulator. The process is blocked until the character is in the accumulator.

The use of this monitor call is not recommended for pseudo-terminals (PTYs). The recommended procedure for reading a character from the PTY output buffer is to open the PTY for input with OPENF and then perform input with the BIN call.

Generates an illegal instruction interrupt on error conditions below.

STO ERROR MNEMONICS:

TTYX1: device is not a terminal

DESX2: terminal is not available to this job

DEVX2: device already assigned to another job

TTYX01: line is not active

STPAR JSYS 217

Sets the device-related modes for the specified terminal. The modes that can be set by this call are in the following bits of the JFN mode word. (Refer to Section 2.4.3.1.)

B1(TT%MFF)	mechanical form feed
B2(TT%TAB)	mechanical tab
B3(TT%LCA)	lower case
B4-B10(TT%LEN)	page length
B11-B17(TT%WID)	page width
B25(TT%ECM)	echo control
B30(TT%UOC)	upper-case output control
B31(TT%LIC)	lower-case input control
B32-B33(TT%DUM)	duplex mode
B34(TT%PGM)	output page mode

ACCEPTS IN AC1: file designator

AC2: JFN mode word

RETURNS +1: always

The STPAR monitor call is a no-op if the designator is not associated with a terminal.

The SFMOD monitor call can be used to set program-related modes of the JFN mode word, and the RFMOD monitor call can be used to obtain the JFN mode word.

When the page length and width fields are set with the STPAR call, they have a maximum range of 127. The MTOPR call can be used to set these fields to values greater than 127. A nonzero value of less than 2 for the length or less than 10 for the width causes STPAR to leave the field unchanged.

STPAR ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX3: JFN is not assigned

DESX5: file is not open

DEVX2: device already assigned to another job

TTYX01: line is not active

STPPN JSYS 556

Translates the given directory name string to its corresponding project-programmer number (a TOPS-10 36-bit directory designator). This project-programmer number is associated with the structure containing the given directory and is valid only for the current mounting of that structure. The STPPN monitor call and the PPNST monitor call should appear only in programs that require translations of project-programmer numbers. Both calls are temporary calls and may not be defined in future releases.

ACCEPTS IN AC1: pointer to ASCIZ string containing the directory name, a JFN, or a 36-bit directory number

RETURNS +1: always, with the corresponding project-programmer number in AC2

Generates an illegal instruction interrupt on error conditions below.

STPPN ERROR MNEMONICS:

STRX02: insufficient system resources

STRX03: no such directory name

STRX04: ambiguous directory specification

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

DESX8: file is not on disk

DESX10: structure is dismounted

STSTS JSYS 25

Sets the status of a file. (Refer to the GTSTS monitor call for the format of the JFN status word.)

ACCEPTS IN AC1: JFN in the right half

AC2: status word

RETURNS +1: failure, error code in AC1

+2: success

The STSTS call is used to set the following bits of the status word:

B9(GS%ERR) file may be in error
B13(GS%HLT) I/O errors are terminating conditions
B17(GS%FRK) file, if opened, is opened for restricted access

STSTS ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

STTYP JSYS 302

Sets the terminal type number for the specified terminal line. (Refer to Section 2.4.3.4.)

ACCEPTS IN AC1: file designator (only terminal designators are legal)

AC2: terminal type number

RETURNS +1: always

The STTYP call sets the bits in the JFN mode word for mechanical form feed and tab, lower case, and page length and width according to their settings in the device characteristics word. These bits can subsequently be changed with the STPAR monitor call.

The GTTYP monitor call can be used to obtain the terminal type number for a specified line.

Generates an illegal instruction interrupt on error conditions below.

STTYP ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

STYPX1: invalid terminal type

TTYX01: line is not active

SWJFN JSYS 47

Swaps the association of two JFNs by literally exchanging all information cells of each JFN.

ACCEPTS IN AC1: JFN

AC2: another JFN

RETURNS +1: always

Generates an illegal instruction interrupt on error conditions below.

SWJFN ERROR MNEMONICS:

DESX1: invalid source/destination designator

DESX2: terminal is not available to this job

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

SWJFX1: illegal to swap same JFN

SYERR JSYS 527

Places information in the System Error (SYSERR) file. (Refer to the DECSYSTEM-20 Operator's Guide for information on the SYSERR file.) The SYERR monitor call requires the process to have WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: address of argument block

AC2: length of argument block

RETURNS +1: always

Generates an illegal instruction interrupt on error conditions below.

SYERR ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required

SYEX1: unreasonable SYSERR block size

SYEX2: no buffer space available for SYSERR

SYSGT JSYS 16

Returns the table number, table length, and word 0 of the specified system table. (Refer to Section 2.3.2 for the names of the system tables.)

ACCEPTS IN AC1: SIXBIT table name

RETURNS +1: always, with

AC1 containing word 0 of the table

AC2 containing the negative of the number of words in the table in the left half, and the table number in the right half

The table number returned can be given to the GETAB monitor call as an argument. However, because the MONSYM file includes symbol definitions for the system tables, execution of the SYSGT call is not required to obtain the table number for the GETAB call.

The contents of AC2 is 0 on return if the specified table was not found.

TBADD JSYS 536

Adds an entry to a standard-formatted command table used for user program command recognition. (Refer to the TBLUK call description for the format of the command table.)

ACCEPTS IN AC1: address of word 0 (header word) of table

AC2: entry to be added to table. (Refer to the TBLUK call for the format of a table entry.)

RETURNS +1: always, address in the table of the new entry in AC1

Generates an illegal instruction interrupt on error conditions below.

TBADD ERROR MNEMONICS:

TADDX1: table is full

TADDX2: entry is already in table

TBDEL JSYS 535

Deletes an entry from a standard-formatted command table used for user program command recognition. (Refer to the TBLUK call description for the format of the command table.)

ACCEPTS IN AC1: address of word 0 (header word) of table

AC2: address of entry to be deleted. This address is returned in AC1 on a TBLUK call.

RETURNS +1: always

Generates an illegal instruction interrupt on error conditions below.

TBDEL ERROR MNEMONICS:

TDELX1: table is empty

TDELX2: invalid table entry location

TBLUK JSYS 537

Compares the specified string in the caller's address space with strings indicated by a standard-formatted command table. This call is used to implement a consistent style of user program command recognition and abbreviation. The TBLUK call performs the function of string lookup in the table, and the TBADD and TBDEL calls perform the functions of adding to and deleting from the table.

The command table has the following format:

Word	Meaning
0	Number of actual entries (not including this entry) in the table in the left half, and maximum number of entries in the table (not including this entry) in the right half.
1 through n	Address of an argument in the left half. This argument contains optional bits pertinent to the string followed by the ASCIZ string itself. The right half of each table entry is available for use by the user program.

The argument pointed to by the left half of each table entry can have one of two formats, depending on the setting of bits 0-7 of the first word of the argument. If bits 0-6 are all off and B7(CM%FW) is on, the string actually begins in the next word of the argument, and the remainder of this word contains data bits relevant to the string.

Table Entry

```
0                        17 18                           35
!=========================================================!
!            ADR             !      for use by program    !
!=========================================================!
```

Argument

```
     0             6 7                                     35
     !=========================================================!
ADR  !       0      !1!              data bits                 !
     !---------------------------------------------------------!
     !                      start of string                    !
     !=========================================================!
```

The following bits are currently defined:

Bit	Symbol	Meaning
34	CM%NOR	Do not recognize this string, even if a string is specified that matches exactly, and consider an exact match as ambiguous. A program can set this bit to include entries that are initial substrings of other entries in the table to enforce a minimum abbreviation of these other entries (e.g., to include D and DE in the table to enforce DEL as the minimum abbreviation of DELETE).
7	CM%FW	Indicate that the remainder of this word is a flag word containing data bits relevant to the string. This bit must be on to distinguish a flag word from a null string.

If any bit of bits 0-6 of the first word of the argument is on or if B7(CM%FW) is off, the string begins in that word. In this case, the data bits do not apply and are assumed to be off.

Table Entry

```
0                        17 18                           35
!=========================================================!
!            ADR             !                            !
!=========================================================!
```

Argument

```
     0                                                     35
     !=========================================================!
ADR  !                      start of string                    !
     !=========================================================!
```

The addresses in the command table must be sorted according to the alphabetical order of the strings. Note that letters are always considered as uppercase. Therefore, the strings ABC and abc are considered equivalent strings. This order results in efficient searching of strings and determination of ambiguous strings.

The right half of each table entry can be used by the program for an address to a dispatch table for the command or for a pointer to a parameter block for additional information about the call. The contents of this half word is ignored by the three table calls.

ACCEPTS IN AC1: address of word 0 (header word) of table

AC2: pointer to string in caller's address space that is to be compared with the string in the table

RETURNS +1: always, with

AC1 containing the address of the entry that matches the input string or address where the entry would be if it were in the table.

AC2 containing recognition bits:

B0(TL%NOM)	The input string does not match any string in the table.
B1(TL%AMB)	The input string matches more than one string in the table (i.e., is ambiguous).
B2(TL%ABR)	The input string is a valid abbreviation of a string in the table.
B3(TL%EXM)	The input string is an exact match with a string in the table.

AC3 containing a pointer to the remainder of the string in the table if the match was on an abbreviation (TL%ABR is on). This string can then be output to complete the command.

Generates an illegal instruction interrupt on error conditions below.

TBLUK ERROR MNEMONICS:

TLUKX1: internal format of table is incorrect

TEXTI JSYS 524

Reads input from a terminal or a file into a string in the caller's address space. Input is read until either a specified break character is encountered or the byte count is exhausted, whichever occurs first.

When used for terminal input, the TEXTI call handles the following editing functions:

1. Delete the last character input (DELETE).

2. Delete back to the last punctuation character (CTRL/W).

3. Delete back to the beginning of the current line or, if the current line is empty, back to the beginning of the previous line (CTRL/U).

4. Retype the current line from its beginning or, if current line is empty, retype the previous line (CTRL/R).

5. Accept the next character without regard to its usual meaning (CTRL/V).

ACCEPTS IN AC1: address of argument block

RETURNS +1: failure, error code in AC1

+2: success, updated pointer in word .RDDBP, appropriate bits set in the left half of word .RDFLG, and updated count in word .RDDBC of the argument block

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.RDCWB	Count of words following this word in the argument block.
1	.RDFLG	Flag bits. (See below.)
2	.RDIOJ	Pointer to string, or input JFN in the left half and output JFN in the right half (if RD%JFN is on in the flag word .RDFLG). The input JFN is where the input is being read from, and the output JFN is where any output generated from character editing is placed.
3	.RDDBP	Pointer to string in caller's address space where input is to be placed (destination string pointer).
4	.RDDBC	Number of bytes available in the destination string.
5	.RDBFP	Pointer to the beginning of the destination buffer. This pointer indicates the maximum limit to which the user can edit back into the buffer with DELETE, CTRL/W, or CTRL/U. This buffer is not separate (i.e., is not disjoint) from the destination string. On the first TEXTI, this pointer is normally the same as the destination string pointer (.RDDBP), but does not have to be the same.
6	.RDRTY	Pointer to the beginning of the prompting-text (CTRL/R) buffer. The text in this buffer, along with any text in the destination buffer, is output if the user types CTRL/R on his first line of input. If this buffer is not set up or the user types CTRL/R on other than the first line of input, only the text in the destination buffer will be output.

6	.RDRTY (Cont.)	The CTRL/R buffer is useful for retyping characters that preceded the user's input, such as a prompt from the program. The text in this buffer cannot be edited by the user, and if the user deletes back to the end of this buffer, his action is treated as if he has deleted all of his input. This buffer is logically adjacent to the destination buffer, but may be physically disjoint from it. When the CTRL/R buffer is disjoint, it must be terminated with a null byte.
7	.RDBRK	Address of a 4-word block of break character mask bits. If a bit is on in the mask, then the corresponding character is considered a break character. Any bits set in this mask override break characters set in the flag word. The mask occupies the leftmost 32 bits of each word, thereby allowing a mask of 128 bits. The rightmost 4 bits of each word are ignored. The mapping is from left to right. The ASCII character set maps into this 128-bit mask. If this word is zero, there is no break character set mask defined.
10	.RDBKL	Pointer to the backup limit in the destination buffer. This pointer indicates the position in the destination buffer to which the user can edit back without being informed. This pointer is used to indicate to the program that previously parsed text has been edited and may need to be reparsed by the program. The pointer can either be equal to the start of the buffer pointer (.RDBFP) or to the destination string pointer (.RDDBP) or be between these two pointers.

Words 5 through 10 (.RDBFP through .RDBKL) in the argument block are optional. A zero in any of the words means that no pointer has been given.

The illustration below is a logical arrangement of the CTRL/R and destination buffers, with the placement of the pointers when they are given as not being equal. Remember that the CTRL/R buffer does not have to be adjacent to the destination buffer and that two or more of these pointers can be equal.

```
                                 destination buffer
                             |                           |
                             |              can be edited|
                             v                           v
!=======================================================!
! CTRL/R buffer;         !  Cannot be    !      !        !
! cannot be edited,      !  edited       !      !        !
! but will be output     !               !      !        !
! on a CTRL/R            !               !      !        !
!=======================================================!
^                        ^               ^      ^
|                        |               |      |
CTRL/R         Beginning of       Backup      Destination
buffer         destination        limit       string
pointer        buffer pointer     pointer     pointer
(.RDRTY)       (.RDBFP)           (.RDBKL)    (.RDDBP)
```

The flag bits that can be set in word 1 (.RDFLG) of the argument block are as follows:

Bit	Symbol	Meaning
0	RD%BRK	Break on CTRL/Z or ESC.
1	RD%TOP	Break on CTRL/G, CTRL/L, CTRL/Z, ESC, carriage return, line feed.
2	RD%PUN	Break on punctuation: CTRL/A-CTRL/F ASCII codes 34-37 CTRL/H-CTRL/I ASCII codes 40-57 CTRL/K ASCII codes 72-100 CTRL/N-CTRL/Q ASCII codes 133-140 CTRL/S-CTRL/T ASCII codes 173-176 CTRL/X-CTRL/Y
3	RD%BEL	Break on end of line (carriage return and line feed, or line feed only).
4	RD%CRF	Suppress a carriage return and return a line feed only.
5	RD%RND	Return to user program if the user tries to delete beyond the beginning of the destination buffer. If this bit is not set, the TEXTI call causes the terminal's bell to ring and waits for more input.
6	RD%JFN	JFNs have been given for the source designator (word .RDIOJ of the argument block). If this bit is not set, the source designator is a pointer to a string.
7	RD%RIE	Return to user program if the input buffer is empty. If this bit is not set, the TEXTI call waits for more input.
8	RD%BBG	Pointer to the beginning of the destination buffer has been given in word .RDBFP in the argument block.

10	RD%RAI	Convert lower-case input to upper-case input.
11	RD%SUI	Suppress the CTRL/U indication if user types a CTRL/U (i.e., do not print XXX and on display terminals, do not delete the characters from the screen).

On a successful return, the following bits can be set in word 1 (.RDFLG) of the argument block:

Bit	Symbol	Meaning
12	RD%BTM	A break character terminated the input. If this bit is not set, the input was terminated because the byte count was exhausted.
13	RD%BFE	Control was returned to the user program because the user tried to delete beyond the beginning of the destination buffer and RD%RND was on in the call.
14	RD%BLR	The backup limit for editing was reached.

TEXTI ERROR MNEMONICS:

RDTX1: invalid string pointer

IOX11: quota exceeded or disk full

TFORK JSYS 321

Sets and removes monitor call intercepts (JSYS traps) for the given inferior processes. When the process attempts to execute a call on which an intercept has been set, it is suspended before it executes the call. Control is passed to the closest superior process that is monitoring the execution of that call. This control is passed by the superior receiving an interrupt when the inferior process is suspended. The superior process can then determine via the RTFRK call which process caused the interrupt and how to handle the interrupt. It can use any of the process manipulation calls and then use the UTFRK call to resume the suspended inferior process. Alternatively, the superior can simply decide to resume the inferior and allow it to execute the call. If this is the case, the next higher superior process monitoring the intercepted call receives an interrupt, and control is passed to that superior. If each superior process monitoring the call decides to resume the suspended process without changing its PC word, then the suspended process is allowed to execute the monitor call as it normally would.

ACCEPTS IN AC1: function code in the left half, and process handle in the right half

AC2: software interrupt channel number in the left half, and size (in bits) of the monitor call bit table

AC3: address of monitor call bit table

RETURN +1: always

The available functions are as follows:

Code	Symbol	Meaning
0	.TFSET	Set monitor call intercepts for the given process. The calls that will be intercepted are indicated in the monitor call bit table. The given process must be frozen.
1	.TFRAL	Remove all monitor call intercepts for the given process. The process must be frozen.
2	.TFRTP	Remove for the given process only the monitor call intercepts that are indicated in the monitor call bit table. The given process must be frozen.
3	.TFSPS	Set the given software channel as the channel on which to generate the interrupt.
4	.TFRPS	Return in the left half of AC2 the software channel on which the interrupt will be generated.
5	.TFTST	Test if the caller is to be intercepted when it attempts to execute monitor calls. On successful return AC2 contains -1 if it is to be intercepted or 0 if it is not to be intercepted.
6	.TFRES	Remove intercepts set for all inferiors and clear the software channel assigned to the interrupt for monitor call intercepts.
7	.TFUUO	Set monitor call intercepts for TOPS-10 monitor calls (UUOs) for the given process. The process must be frozen.
8	.TFSJU	Set monitor call intercepts for both the calls indicated in the monitor call bit table and the TOPS-10 monitor calls. This function is a combination of functions .TFSET and .TFUUO. The given process must be frozen.
9	.TFRUU	Remove monitor call intercepts for the TOPS-10 monitor calls. The given process must be frozen.

The process handle in the right half of AC1 must refer to an inferior process or must be -4 to refer to all inferiors. When intercepts are set for a given process, they also apply to all processes inferior to the given process. When a process is created, it is subject to the same intercepts as the process that created it.

If the software channel is given as 77, any intercepts bypass the given process without causing either an interrupt to its superior or a suspended state of the process.

The monitor call bit table contains a bit for each of the TOPS-20 monitor calls. When a bit in the table is on, the corresponding monitor call is to be intercepted when the given process attempts to execute it. If the bit is off, the corresponding monitor call will not be intercepted. The size of the bit table is 1000(octal) words.

A process can remove only the intercepts it previously set; it cannot remove intercepts that other processes set.

Refer to the TOPS-20AN Monitor Calls User's Guide for an overview and description of the monitor call intercept facility.

Generates an illegal instruction interrupt on error conditions below.

TFORK ERROR MNEMONICS:

TFRKX1: invalid function code

TFRKX2: unassigned process handle or not immediate inferior

TFRKX3: process not frozen

THIBR JSYS 770

Blocks the current process for the specified elapsed time or until awakened by a TWAKE monitor call, whichever occurs first. The THIBR call is a temporary call and may not be defined in future releases.

ACCEPTS IN AC1: 0 in the left half, and maximum number of seconds to block in the right half

RETURNS +1: never

+2: always, time expired or TWAKE call occurred

TIME JSYS 14

Returns the amount of time since the system was last restarted.

RETURNS +1: always, time (in milliseconds) right-justified in AC1, and divisor to convert the time to seconds in AC2. AC2 always contains 1000; thus, it is not necessary to examine its contents.

This is a monotonically increasing number (when the system is running) independent of any resets of the time and date.

TIMER JSYS 522

Controls the amount of time either a process within a job or the entire job can run. An interrupt is generated when the time has elapsed.

Only one process in the job is allowed to time the entire job. If the job is already being timed, an error is given if another process attempts to time the job. An error is also given if a process other than the one that set the runtime limit of the job attempts to remove that limit.

ACCEPTS IN AC1: process handle in the left half, and function code in the right half.

AC2: time at which to generate an interrupt. Refer to the individual function descriptions for the specific arguments.

AC3: number of the software channel on which to generate an interrupt when the time has expired.

RETURNS +1: failure, error code in AC1

+2: success

The available functions are as follows:

Code	Symbol	Meaning
0	.TIMRT	Specify the total runtime of the entire job. This function allows one process within a job to time the entire job. AC2 contains the total runtime in milliseconds that the job can accumulate before an interrupt is generated on the specified channel. If AC2 contains 0, the limit on the runtime of the job is removed. The process handle given in AC1 must be .FHJOB (-5).
1	.TIMEL	Specify an elapsed time after which an interrupt is generated for the given process. AC2 contains the number of milliseconds that can now elapse before the interrupt is generated on the specified channel.
2	.TIMDT	Specify an exact time at which an interrupt is generated for the given process. AC2 contains the intermal format (refer to section 2.8.2) of the date and time when the interrupt is to be generated.
3	.TIMDD	Remove any pending interrupt requests that are to occur for the process at the given time. AC2 contains the internal format (refer to section 2.8.2) of the date and time of the interrupt request to be removed.

4	.TIMBF	Remove any pending interrupt requests that are to occur for the process before the given time. AC2 contains the internal format (refer to section 2.8.2) of the date and time.
5	.TIMAL	Remove all pending requests for the given process including the runtime limit on the entire job.

The runtime limit for a job can be obtained via the GETJI monitor call (contents of word .JIRT on return). If the job's time limit has been exceeded, the value returned by the GETJI call will be zero.

TIMER ERROR MNEMONICS:

TIMX1: invalid function

TIMX2: invalid process handle

TIMX3: time limit already set

TIMX4: illegal to clear time limit

TIMX5: invalid software interrupt channel number

TIMX6: time has already passed

TIMX7: no space available for a clock

TIMX8: user clock allocation exceeded

TIMX9: no such clock entry found

TIMX10: no system date and time

TLINK JSYS 216

Controls terminal linking. (Refer to Section 2.4.3.5 for more information.)

ACCEPTS IN AC1:	B0(TL%CRO)	Clear link from remote to object designator. If the remote designator is -1, all remote links to the object designator are cleared.
	B1(TL%COR)	Clear link from object to remote designator. If the remote designator is -1, links from the object to all remote designators are cleared.
	B2(TL%EOR)	Establish link from object to remote designator.
	B3(TL%ERO)	Establish link from remote to object designator.

B4(TL%SAB)	Examine B5(TL%ABS) to determine the setting of the object designator's accept link bit. If this bit is off, B5 is ignored.
B5(TL%ABS)	Set the object designator's accept link bit. When B4(TL%SAB) is on, the object designator is accepting links if TL%ABS is on and refusing links if TL%ABS is off.
B6(TL%STA)	Examine B7(TL%AAD) to determine the setting of the object designator's accept advice bit. If this bit is off, B7 is ignored.
B7(TL%AAD)	Set the object designator's accept advice bit. When B6(TL%STA) is on, the object designator is accepting advice if TL%AAD is on and refusing advice if TL%ADD is off.
B18-B35 (TL%OBJ)	Object designator

AC2: remote designator

RETURNS +1: failure, error code in AC1

+2: success

The object and remote designators must be either 4xxxxx or -1. An object designator of -1 indicates the controlling terminal.

The following restrictions apply if the process does not have WHEEL capability enabled:

1. The object designator must specify a terminal assigned to this job.

2. The object-to-remote link must be specified before or at the same time as the remote-to-object link.

If the accept bit of the remote designator is not set, a link from the object-to-remote designator causes the remote designator's bell to ring. If the remote designator does not set the accept bit within 15 seconds, the TLINK call returns an error.

When terminals are linked together and a character is typed on one terminal, the same ASCII character code is sent to all terminals in the link. The character always appears in the output buffers of all terminals regardless of the current mode of each individual terminal. The character is sent according to the data mode and terminal type of the terminal that originates the character. For example, if one terminal originates a TAB and has mechanical tabs set, all terminals in the link receive the ASCII code for a TAB in their output buffers.

TLINK ERROR MNEMONICS:

DESX1: invalid source/destination designator

TLNKX1: illegal to set remote to object before object to remote

TLNKX2: link was not received within 15 seconds

TLNKX3: links full

TTYX01: line is not active

TMON JSYS 7

Tests various monitor flags.

ACCEPTS IN AC1: function code

RETURNS +1: always, value of the function in AC2

The codes for the functions are as follows:

Code	Symbol	Meaning
0	.SFFAC	FACT files entries are allowed.
1	.SFCDE	CHECKD found errors.
2	.SFCDR	CHECKD is running.
3	.SFMST	Manual start is in progress.
4	.SFRMT	Remote LOGINS (dataset lines) are allowed.
5	.SFPTY	PTY LOGINS are allowed.
6	.SFCTY	CTY LOGINS are allowed.
7	.SFOPR	Operator is in attendance.
10	.SFLCL	Local LOGINS (hardwired lines) are allowed.
11	.SFBTE	Bit table errors found on startup.
12	.SFCRD	Users can change nonprivileged directory parameters with the CRDIR monitor call.
13	.SFNVT	ARPANET terminal LOGINs are allowed.
21	.SFUSG	USAGE file entries are allowed.
22	.SFFLO	Disk latency optimization using the RH20 backup register is enabled.
44	.SFNTN	ARPANET is on.
45	.SFNDU	ARPANET will be reinitialized if it is down.
46	.SFNHI	ARPANET host table will be initialized.
47	.SFTMZ	Local time zone is set.
50	.SFLHN	Local ARPANET host number is set.
51	.SFAVR	Account validation is running on this system.

Functions 0 through 22 represent a specific monitor flag bit. When the value of the function returned in AC2 is 1, the flag corresponding to the function is set. When the value returned is 0, the flag is not set.

The SMON monitor call can be used to set various monitor flags.

Generates an illegal instruction interrupt on error conditions below.

TMON ERROR MNEMONICS:

TMONX1: invalid TMON function

TTMSG JSYS 775

Sends a message to a specified terminal or to all terminals. This monitor call requires the process to have WHEEL or OPERATOR capability enabled. The TTMSG call is a temporary call and may not be defined in future releases.

ACCEPTS IN AC1: 400000 + TTY number, or -1 to send to all terminals

AC2: pointer to string in caller's address space to be sent

RETURNS +1: always

The TTMSG monitor call is a no-op if the specified terminal does not exist.

Generates an illegal instruction interrupt on error conditions below.

TTMSG ERROR MNEMONICS:

GTDIX1: WHEEL or OPERATOR capability required

TWAKE JSYS 771

Wakes the specified job that is blocked because of the execution of a THIBR call. If more than one process in a job is blocked because of a THIBR call, execution of the TWAKE call causes any one of the processes to be awakened. The TWAKE call is a temporary call and may not be defined in future releases.

ACCEPTS IN AC1: 0 in the left half, and number of job to be awakened in the right half

RETURNS +1: failure, error code in AC1

+2: success, signal sent. Job will be awakened immediately if blocked by a THIBR call or as soon as next THIBR call is executed.

TWAKE ERROR MNEMONICS:

ATACX1: invalid job number

UFPGS JSYS 525

Updates pages of the specified file. This monitor call is used to guarantee that a certain sequence of file pages has been written to the disk before any other operation is performed.

ACCEPTS IN AC1: JFN in the left half, and file page number of the first page to be updated in the right half

AC2: count of number of sequential pages to update

RETURNS +1: failure, error code in AC1

+2: success, all modified pages are written to disk. The FDB is updated, if necessary.

The UFPGS call causes the process to block until all writes to the disk are completed.

UFPGS ERROR MNEMONICS:

UFPGX1: file is not opened for write

DESX3: JFN is not assigned

DESX4: invalid use of terminal designator or string pointer

DESX7: JFN cannot refer to output wildcard designators

DESX8: file is not on disk

LNGFX1: page table does not exist and file not open for write

IOX11: quota exceeded or disk full

USAGE JSYS 564

Controls accounting on the system by writing entries into the system's data file. All entries to the data file are made with this call. Examples of the types of entries entered into the data file are disk storage usage, input and output spooler usage, job session entry, and date and time changes.

The file written by the USAGE call is an intermediate binary file, which is converted by a system program to the final ASCII file. Each entry in the final file is at least two records long, each record being defined as a string of ASCII characters terminated with a line-feed character. The first record contains system and file information; its format is the same for all entries. Subsequent records contain data pertaining to the entry; their formats vary according to the particular data being entered.

The USAGE monitor call requires the process to have WHEEL or OPERATOR capability enabled.

Refer to the DECSYSTEM-20 USAGE File Specification for additional information on the system's data file.

ACCEPTS IN AC1: function code

AC2: function argument or address of record descriptor block

RETURNS +1: always

The available functions are as follows:

Code	Symbol	Meaning
0	.USENT	Write an entry into the system's data file. AC2 contains the address of the record descriptor block.
1	.USCLS	Close the system's data file, which is named PS:<ACCOUNTS>SYSTEM-DATA.BIN. No additional entires are recorded into this file and a new SYSTEM-DATA.BIN is opened for subsequent entries.
2	.USCKP	Perform a checkpoint of all jobs. Data recorded during a checkpoint includes the billable data (e.g., connect time, runtime) accumulated during the job session, starting from time of login or the last SET ACCOUNT command and ending at the time this function is performed. The data collected on a LOGIN or SET ACCOUNT command is entered into the session entry in the data file. The default checkpoint time interval is 10 minutes.
3	.USLGI	Initialize a checkpoint entry for the job. This function is used internally by the LOGIN monitor call. AC2 contains the address of the record descriptor block.
4	.USLGO	Terminate the checkpoint entry for the job and write an entry into the system's data file, which is named PS:<ACCOUNTS>SYSTEM-DATA.BIN. This function is used internally by the LGOUT monitor call. AC2 contains the address of the record descriptor block.
5	.USSEN	Terminate the current session, write an entry into the system's data file, which is named PS:<ACCOUNTS>SYSTEM-DATA.BIN, and initialize a new checkpoint entry for the job. This function is used internally by the CACCT monitor call. AC2 contains the address of the record descriptor block.
6	.USCKI	Set the checkpoint time interval. AC2 contains the interval in minutes.
7	.USENA	Install the accounting data base from the file named PS:<SYSTEM>ACCOUNTS-TABLE.BIN into the running monitor. The ACTGEN program uses this file to generate the list of valid accounts.

The record descriptor block, whose address is given in AC2, is set up by the UITEM. macro defined in ACTSYM.MAC. (Refer to Appendix B for the definition of the UITEM. macro.) The names of all data entries are generated by this macro. The USENT. macro is used to generate the header of the record descriptor block.

The format of the data generated by the USAGE call is a list of items describing the entries in a single record. This list has a header word containing the version numbers and the type of entry. The data words follow this header with two words per data item. The list is terminated with a zero word.

Generates an illegal instruction interrupt on error conditions below.

USAGE ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required

ARGX02: invalid function

USGX01: invalid USAGE entry type code

USGX02: item not found in argument list

USGX03: default item not allowed

USRIO JSYS 310

Places the user program into user I/O mode in order that it can execute various hardware I/O instructions. The user IOT flag is turned on in the PC of the running process. The program can leave user I/O mode by executing a JRSTF (e.g., JRSTF @[.+1]).

RETURNS +1: failure, error code in AC1

+2: success, user IOT flag is set

The USRIO monitor call requires that the process have WHEEL, OPERATOR, or MAINTENANCE capability enabled.

USRIO ERROR MNEMONICS:

CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required

UTEST JSYS 563

Provides a method for determining if every instruction in a section of monitor code actually gets executed. This monitor call does not test the code by executing it; it confirms that a test of the code is complete by reporting the instructions that were executed during the test.

This monitor call requires the process to have WHEEL capability enabled.

ACCEPTS IN AC1: function code in the left half, and length of the argument block in the right half.

AC2: address of the argument block

RETURNS +1: always

The available functions are as follows:

Code	Symbol	Meaning
0	.UTSET	Start testing of the code.
1	.UTCLR	Stop testing of the code and update the bit map in the argument block.

The format of the argument block is as follows:

Word	Symbol	Meaning
0	.UTADR	Address of the beginning of the section of code that is to be tested.
1	.UTLEN	Length of section of code that is to be tested.
2	.UTMAP	Start of bit map representing the instructions that are to be tested in the section of code. This map contains one bit for each location in the section. If a bit is on in the map, the corresponding instruction is to be tested. If a bit is off, the corresponding instruction is not to be tested. Locations that contain data and that would cause the section of code to execute improperly if that data were changed should not be tested.

Internally, a copy of the code being tested is placed in a buffer, which is dynamically locked down during execution of the UTEST call. The system allows any monitor routine to be tested as long as a pushdown stack pointed to by AC P (i.e., 17) is set up whenever the routine is called.

After execution of the .UTCLR function, the bit map is changed to reflect the instructions that were actually executed during the test. If a bit is on in the map, the corresponding instruction was executed. If a bit is off, the corresponding instruction was not executed.

Generates an illegal instruction interrupt on error conditions below.

UTEST ERROR MNEMONICS:

CAPX3:	WHEEL capability required
UTSTX1:	invalid function code
UTSTX2:	area of code too large to test
UTSTX3:	UTEST facility in use by another process

UTFRK JSYS 323

Resumes the execution of a process that is suspended because of a monitor call intercept. The instruction where the execution resumes depends on the current PC word of the suspended process. To prevent the suspended process from executing the call, the superior process handling the intercept can change the PC word (via the SFORK or SFRKV call). Then on execution of the UTFRK call, the suspended process continues at the new PC. If the superior process handling the intercept does not change the PC word of the suspended process, then the next superior process intercepting that particular monitor call will receive the interrupt.

Refer to the TOPS-20AN Monitor Calls User's Guide for an overview and description of the monitor call intercept facility.

ACCEPTS IN AC1: flag bits in the left half, and process handle in the right half

RETURNS +1: always

The flag bit that can be given in AC1 is as follows:

Bit	Symbol	Meaning
0	UT%TRP	Cause a failure return for the suspended process. This return will be either the generation of an illegal instruction interrupt or the processing of an ERJMP or ERCAL instruction.

The UTFRK monitor call is a no-op if

1. The process handle given is valid but the process specified is not suspended because of a monitor call intercept.

2. The caller is not one of the processes monitoring the suspended process and therefore is not permitted to resume the process.

Generates an illegal instruction interrupt on error conditions below.

UTFRK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

FRKHX3: invalid use of multiple process handle

VACCT JSYS 566

Verifies accounts by validating the supplied account for the given user.

ACCEPTS IN AC1: 36-bit user number, 36-bit directory number, or -1 to validate the account for the current user

AC2: pointer to account string

RETURNS +1: always, with updated pointer in AC2

Generates an illegal instruction interrupt on error conditions below.

VACCT ERROR MNEMONICS:

VACCX0: invalid account

VACCX1: account string exceeds 39 characters

VACCX2: account has expired

MONX02: insufficient system resources (JSB full)

DELFX6: internal format of directory is incorrect

DIRX1: invalid directory number

DIRX3: internal format of directory is incorrect

STRX01: structure is not mounted

OPNX9: invalid simultaneous access

OPNX16: file has bad index block

WAIT JSYS 306

Dismisses the current process indefinitely and does not return. If the software interrupt system is enabled for this process, the process can be interrupted out of the wait state. Upon execution of a DEBRK call, the process continues to wait until the next interrupt unless the interrupt routine changes the PC word. In this case, the process resumes execution at the new PC location. If the interrupt routine changes the PC word, it must set the user-mode bit (bit 5) of the PC word. (Refer to Section 2.5.7.)

WFORK JSYS 163

Causes the current process to wait for an inferior process to terminate (voluntarily or involuntarily).

ACCEPTS IN AC1:	process handle, or -4 in the right half to wait for any one of the inferior processes to terminate
RETURNS +1:	always, when one of the specified processes terminates

This call returns immediately if the specified process or one of the inferior processes has already terminated.

Generates an illegal instruction interrupt on error conditions below.

WFORK ERROR MNEMONICS:

FRKHX1: invalid process handle

FRKHX2: illegal to manipulate a superior process

APPENDIX A

MONSYM.MAC

This appendix contains the complete copy of the system file MONSYM.MAC, which defines the symbols used in the manual. The user must include the statement

```
SEARCH MONSYM
```

in his program to have the symbols defined in his assembly.

```
;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED
;  OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.
;
;COPYRIGHT (C) 1976, 1977, 1978 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

;MONITOR CALL DEFINITIONS AND ERROR MNEMONICS

;NOTE:
;       THE FOLLOWING SYMBOLS ARE RESERVED:
;
;               SYMBOL          RESERVED BY
;               ======          ===========
;
;               .OF???          RMS-20
;               .SZ???          RMS-20
;               .PS???          RMS-20

;MACRO TO DEFINE JSYS NAMES

DEFINE DEFJS (NAME,NUM,SECT,XTRA)<
        OPDEF NAME [104B8+NUM]
        IFDEF .PSECT,<
        INTERN NAME>>

        SALL

IFNDEF REL,<REL==0>             ;ASSEMBLING REL IF NON-0
  IFE REL,<
        UNIVERSAL MONSYM>
   IFN REL,<
        TITLE MONSYM
        IFNDEF .PSECT,<
        .DIRECT .XTABM>
        >
```

```
;JSYS DEFINITIONS WITH 'NIM' AS A FOURTH ARGUMENT ARE CLASSIFIED
; AS 'NOT IN MONITOR'

DEFINE JSLIST <

DEFJS JSYS,0,,NIM

DEFJS LOGIN,1,MSEC1
DEFJS CRJOB,2,MSEC1
DEFJS LGOUT,3,MSEC1
DEFJS CACCT,4,MSEC1
DEFJS EFACT,5,MSEC1
DEFJS SMON,6,MSEC1
DEFJS TMON,7,MSEC1
DEFJS GETAB,10,MSEC1
DEFJS ERSTR,11,MSEC1
DEFJS GETER,12,MSEC1
DEFJS GJINF,13,MSEC1
DEFJS TIME,14,MSEC1
DEFJS RUNTM,15,MSEC1
DEFJS SYSGT,16,MSEC1
DEFJS GNJFN,17,MSEC1
DEFJS GTJFN,20,MSEC1
DEFJS OPENF,21,MSEC1
DEFJS CLOSF,22,MSEC1
DEFJS RLJFN,23,MSEC1
DEFJS GTSTS,24,MSEC1
DEFJS STSTS,25,MSEC1
DEFJS DELF,26,MSEC1
DEFJS SFPTR,27,MSEC1
DEFJS JFNS,30,MSEC1
DEFJS FFFFP,31,MSEC1
DEFJS RDDIR,32,MSEC1        ;OBSOLETE
DEFJS CPRTF,33,,NIM
DEFJS CLZFF,34,MSEC1
DEFJS RNAMF,35,MSEC1
DEFJS SIZEF,36,MSEC1
DEFJS GACTF,37,MSEC1
DEFJS STDIR,40,MSEC1        ;OBSOLETE
DEFJS DIRST,41,MSEC1
DEFJS BKJFN,42,MSEC1
DEFJS RFPTR,43,MSEC1
DEFJS CNDIR,44,,NIM
DEFJS RFBSZ,45,MSEC1
DEFJS SFBSZ,46,MSEC1
DEFJS SWJFN,47,MSEC1
DEFJS BIN,50,MSEC1
DEFJS BOUT,51,MSEC1
DEFJS SIN,52,MSEC1
DEFJS SOUT,53,MSEC1
DEFJS RIN,54,MSEC1
DEFJS ROUT,55,MSEC1
DEFJS PMAP,56,MSEC1
DEFJS RPACS,57,MSEC1
DEFJS SPACS,60,MSEC1
```

```
DEFJS RMAP,61,MSEC1
DEFJS SACTF,62,MSEC1
DEFJS GTFDB,63,MSEC1
DEFJS CHFDB,64,MSEC1
DEFJS DUMPI,65,MSEC1
DEFJS DUMPO,66,MSEC1
DEFJS DELDF,67,MSEC1
DEFJS ASND,70,MSEC1
DEFJS RELD,71,MSEC1
DEFJS CSYNO,72,,NIM
DEFJS PBIN,73,MSEC1
DEFJS PBOUT,74,MSEC1
DEFJS PSIN,75,,NIM
DEFJS PSOUT,76,MSEC1
DEFJS MTOPR,77,MSEC1
DEFJS CFIBF,100,MSEC1
DEFJS CFOBF,101,MSEC1
DEFJS SIBE,102,MSEC1
DEFJS SOBE,103,MSEC1
DEFJS DOBE,104,MSEC1
DEFJS GTABS,105,MSEC1     ;OBSOLETE
DEFJS STABS,106,MSEC1     ;OBSOLETE
DEFJS RFMOD,107,MSEC1
DEFJS SFMOD,110,MSEC1
DEFJS RFPOS,111,MSEC1
DEFJS RFCOC,112,MSEC1
DEFJS SFCOC,113,MSEC1
DEFJS STI,114,MSEC1
DEFJS DTACH,115,MSEC1
DEFJS ATACH,116,MSEC1
DEFJS DVCHR,117,MSEC1
DEFJS STDEV,120,MSEC1
DEFJS DEVST,121,MSEC1
DEFJS MOUNT,122,MSEC1     ;OBSOLETE
DEFJS DSMNT,123           ;OBSOLETE
DEFJS INIDR,124,MSEC1     ;OBSOLETE
DEFJS SIR,125,MSEC1
DEFJS EIR,126,MSEC1
DEFJS SKPIR,127,MSEC1
DEFJS DIR,130,MSEC1
DEFJS AIC,131,MSEC1
DEFJS IIC,132,MSEC1
DEFJS DIC,133,MSEC1
DEFJS RCM,134,MSEC1
DEFJS RWM,135,MSEC1
DEFJS DEBRK,136,MSEC1
DEFJS ATI,137,MSEC1
DEFJS DTI,140,MSEC1
DEFJS CIS,141,MSEC1
DEFJS SIRCM,142,MSEC1
DEFJS RIRCM,143,MSEC1
DEFJS RIR,144,MSEC1
DEFJS GDSTS,145,MSEC1
DEFJS SDSTS,146,MSEC1
DEFJS RESET,147,MSEC1
```

```
DEFJS RPCAP,150,MSEC1
DEFJS EPCAP,151,MSEC1
DEFJS CFORK,152,MSEC1
DEFJS KFORK,153,MSEC1
DEFJS FFORK,154,MSEC1
DEFJS RFORK,155,MSEC1
DEFJS RFSTS,156,MSEC1
DEFJS SFORK,157,MSEC1
DEFJS SFACS,160,MSEC1
DEFJS RFACS,161,MSEC1
DEFJS HFORK,162,MSEC1
DEFJS WFORK,163,MSEC1
DEFJS GFRKH,164,MSEC1
DEFJS RFRKH,165,MSEC1
DEFJS GFRKS,166,MSEC1
DEFJS DISMS,167,MSEC1
DEFJS HALTF,170,MSEC1
DEFJS GTRPW,171,MSEC1
DEFJS GTRPI,172,MSEC1
DEFJS RTIW,173,MSEC1
DEFJS STIW,174,MSEC1
DEFJS SOBF,175,MSEC1
DEFJS RWSET,176,MSEC1
DEFJS GETNM,177,MSEC1
DEFJS GET,200,MSEC1
DEFJS SFRKV,201,MSEC1
DEFJS SAVE,202,MSEC1
DEFJS SSAVE,203,MSEC1
DEFJS SEVEC,204,MSEC1
DEFJS GEVEC,205,MSEC1
DEFJS GPJFN,206,MSEC1
DEFJS SPJFN,207,MSEC1
DEFJS SETNM,210,MSEC1
DEFJS FFUFP,211,MSEC1
DEFJS DIBE,212,MSEC1
DEFJS FDFRE,213,,NIM
DEFJS GDSKC,214,MSEC1
DEFJS LITES,215,MSEC1        ;OBSOLETE
DEFJS TLINK,216,MSEC1
DEFJS STPAR,217,MSEC1
DEFJS ODTIM,220,MSEC1
DEFJS IDTIM,221,MSEC1
DEFJS ODCNV,222,MSEC1
DEFJS IDCNV,223,MSEC1
DEFJS NOUT,224,MSEC1
DEFJS NIN,225,MSEC1
DEFJS STAD,226,MSEC1
DEFJS GTAD,227,MSEC1
DEFJS ODTNC,230,MSEC1
DEFJS IDTNC,231,MSEC1
DEFJS FLIN,232,MSEC1
DEFJS FLOUT,233,MSEC1
DEFJS DFIN,234,MSEC1
DEFJS DFOUT,235,MSEC1
```

```
DEFJS CRDIR,240,MSEC1
DEFJS GTDIR,241,MSEC1
DEFJS DSKOP,242,MSEC1
DEFJS SPRIW,243,MSEC1
DEFJS DSKAS,244,MSEC1
DEFJS SJPRI,245,MSEC1
DEFJS STO,246,MSEC1

DEFJS ASNDP,260,,NIM
DEFJS RELDP,261,,NIM
DEFJS ASNDC,262,,NIM
DEFJS RELDC,263,,NIM
DEFJS STRDP,264,,NIM
DEFJS STPDP,265,,NIM
DEFJS STSDP,266,,NIM
DEFJS RDSDP,267,,NIM
DEFJS WATDP,270,,NIM

DEFJS ATNVT,274,MSEC1           ;TOPS20AN
DEFJS CVSKT,275,MSEC1           ;TOPS20AN
DEFJS CVHST,276,MSEC1           ;TOPS20AN
DEFJS FLHST,277,MSEC1           ;TOPS20AN

DEFJS GCVEC,300,MSEC1
DEFJS SCVEC,301,MSEC1
DEFJS STTYP,302,MSEC1
DEFJS GTTYP,303,MSEC1
DEFJS BPT,304,MSEC1       ;OBSOLETE
DEFJS GTDAL,305,MSEC1
DEFJS WAIT,306,MSEC1
DEFJS HSYS,307,MSEC1
DEFJS USRIO,310,MSEC1
DEFJS PEEK,311,MSEC1
DEFJS MSFRK,312,MSEC1
DEFJS ESOUT,313,MSEC1
DEFJS SPLFK,314,MSEC1
DEFJS ADVIS,315,,NIM
DEFJS JOBTM,316,,NIM
DEFJS DELNF,317,MSEC1
DEFJS SWTCH,320,MSEC1     ;OBSOLETE
DEFJS TFORK,321,MSEC1
DEFJS RTFRK,322,MSEC1
DEFJS UTFRK,323,MSEC1
DEFJS SCTTY,324,MSEC1

DEFJS SETER,336,MSEC1
```

```
;NEW (NOT IN BBN TENEX) JSYS'S ADDED STARTING AT 500

DEFJS RSCAN,500,MSEC1
DEFJS HPTIM,501,MSEC1
DEFJS CRLNM,502,MSEC1
DEFJS INLNM,503,MSEC1
DEFJS LNMST,504,MSEC1
DEFJS RDTXT,505,MSEC1    ;OBSOLETED BY RDTTY AND TEXTI
DEFJS SETSN,506,MSEC1
DEFJS GETJI,507,MSEC1
DEFJS MSEND,510,MSEC1
DEFJS MRECV,511,MSEC1
DEFJS MUTIL,512,MSEC1
DEFJS ENQ,513,MSEC1
DEFJS DEQ,514,MSEC1
DEFJS ENQC,515,MSEC1
DEFJS SNOOP,516,MSEC1
DEFJS SPOOL,517,MSEC1
DEFJS ALLOC,520,MSEC1
DEFJS CHKAC,521,MSEC1
DEFJS TIMER,522,MSEC1
DEFJS RDTTY,523,MSEC1
DEFJS TEXTI,524,MSEC1
DEFJS UFPGS,525,MSEC1
DEFJS SFPOS,526,MSEC1
DEFJS SYERR,527,MSEC1
DEFJS DIAG,530,MSEC1
DEFJS SINR,531,MSEC1
DEFJS SOUTR,532,MSEC1
DEFJS RFTAD,533,MSEC1
DEFJS SFTAD,534,MSEC1
DEFJS TBDEL,535,MSEC1
DEFJS TBADD,536,MSEC1
DEFJS TBLUK,537,MSEC1
DEFJS STCMP,540,MSEC1
DEFJS SETJB,541,MSEC1
DEFJS GDVEC,542,MSEC1
DEFJS SDVEC,543,MSEC1
DEFJS COMND,544,MSEC1
DEFJS PRARG,545,MSEC1
DEFJS GACCT,546,MSEC1
DEFJS LPINI,547,MSEC1
DEFJS GFUST,550,MSEC1
DEFJS SFUST,551,MSEC1
DEFJS ACCES,552,MSEC1
DEFJS RCDIR,553,MSEC1
DEFJS RCUSR,554,MSEC1
DEFJS MSTR,555,MSEC1
DEFJS STPPN,556,MSEC1
DEFJS PPNST,557,MSEC1
DEFJS PMCTL,560,MSEC1
DEFJS LOCK,561,,NIM
DEFJS BOOT,562,MSEC1
DEFJS UTEST,563
DEFJS USAGE,564,MSEC1
```

```
; HOLE - SLOT 565 AVAILABLE

DEFJS VACCT,566,MSEC1
DEFJS NODE,567,MSEC1
DEFJS ADBRK,570,MSEC1

;TEMPORARY JSYS DEFINITIONS

DEFJS SNDIM,750,MSEC1          ;TOPS20AN
DEFJS RCVIM,751,MSEC1          ;TOPS20AN
DEFJS ASNSQ,752,MSEC1          ;TOPS20AN
DEFJS RELSQ,753,MSEC1          ;TOPS20AN

DEFJS THIBR,770,MSEC1
DEFJS TWAKE,771,MSEC1
DEFJS MRPAC,772,MSEC1
DEFJS SETPV,773,,NIM
DEFJS MTALN,774,MSEC1
DEFJS TTMSG,775,MSEC1

>  ;;; END OF DEFINE JSLIST

;NOW EXPAND THE JSYS DEFINITIONS

JSLIST
```

```
;ERROR CONDITION INSTRUCTIONS.  THESE ARE NOP'S UNLESS  IMMEDIATELY
;FOLLOWING A JSYS WHICH FAILS.

OPDEF ERJMP [JUMP 16,0]          ;JUMP ON ERROR
OPDEF ERCAL [JUMP 17,0]          ;CALL ON ERROR (SIMULATE PUSHJ 17,ADR)
   IFNDEF FOR,<
   IFDEF .PSECT,<
INTERN ERJMP,ERCAL
   >>

DEFINE GOPDEF (OP,DEF)<
        OPDEF OP [DEF]
   IFNDEF FOR,<
   IFDEF .PSECT,<
        INTERN OP>>>

; THE FOLLOWING OPCODES ARE USED TO PERFORM THE EXTENDED
; ADDRESSING FUNCTIONS.

GOPDEF XJRSTF,<JRST 5,0>         ;RESTORE FLAGS AND PC
GOPDEF XJEN,<JRST 6,0>           ;RESTORE FLAGS,PC AND DISMISS
GOPDEF XPCW,<JRST 7,0>           ;EXCHANGE FLAGS AND PC
GOPDEF XSFM,<JRST 14,0>          ;SAVE PC FLAGS IN MEMORY
GOPDEF XMOVEI,<SETMI 0,0>        ;EXTENDED MOVEI
GOPDEF XHLLI,<XMOVEI 0,0>        ;INSTRUCTION TO PUT IMMEDIATE ADDRESS IN LH

DEFINE XBLT (A)<
        EXTEND A,[020000,,0]>

;THE NO-OPERATION INSTRUCTION (MAY CHANGE FROM PROCESSOR TO PROCESSOR)

GOPDEF NOP,<TRN 0,0>

;SPECIAL LOSEG SYMBOL FOR PAT

.JBHSO==:75                      ; 0 ,, HIGHSEG ORIGIN PAGE NUMBER
```

```
;*******************************************
;JSYS SPECIFIC ARGUMENTS
;THE FOLLOWING ARE ORDERED ALPHABETICALLY BY JSYS NAME
;*******************************************


;ACCES - ACCESS A DIRECTORY (E.G., BY CONNECTING)

AC%CON==:1B0                    ;CONNECT TO THE SPECIFIED DIRECTORY
AC%OWN==:1B1                    ;GAIN OWNERSHIP
AC%REM==:1B2                    ;REMOVE OWNERSHIP

;OFFSETS IN ARGUMENT BLOCK

.ACDIR==:0                      ;DIRECTORY DESIGNATOR
.ACPSW==:1                      ;POINTER TO PASSWORD STRING
.ACJOB==:2                      ;JOB NUMBER (-1 FOR SELF)


;ADBRK - Address break JSYS function codes and bits

;FUNCTION CODES

.ABSET==:0                      ;SET USER ADDRESS BREAK
.ABRED==:1                      ;READ USER ADDRESS BREAK
.ABCLR==:2                      ;CLEAR USER ADDRESS BREAK
.ABGAD==:3                      ;GET ADDRESS OF TRAPPED INSTRUCTION

;FUNCTION BITS FOR FUNCTION .ABSET

AB%RED==:1B0                    ;READ
AB%WRT==:1B1                    ;WRITE
AB%XCT==:1B2                    ;EXECUTE


;ALLOC JSYS FUNCTION CODES

.ALCAL==:0                      ;ALLOCATE A DEVICE


;ATNVT               ;TOPS20AN

AN%NTP==:1B2         ;TOPS20AN ;NEW TELNET PROTOCOL


;ATACH

AT%CCJ==:1B0                    ;^C JOB WHEN ATTACHED
AT%NAT==:1B1                    ;NO ATTACH
AT%TRM==:1B2                    ;ATTACH JOB TO TERMINAL IN REGISTER 4
AT%JOB==:777777B35              ;JOB NUMBER
```

```
;BOOT

.BTROM==:0                              ;ACTIVATE ROM BOOT
        .BTDTE==:0                      ;DTE-20 NUMBER
.BTLDS==:1                              ;LOAD SECONDARY BOOTSTRAP PROGRAM
        .BTERR==:1                      ;ERROR FLAGS
        .BTSEC==:2                      ;ADDRESS OF SECONDARY BOOTSTRAP PROGRAM
.BTLOD==:2                              ;LOAD MEMORY
        .BTFLG==:3                      ;FLAGS
                BT%BEL==:1B0            ;SEND TO -11 DOORBELL AFTER SETUP
        .BTCNT==:4                      ;NUMBER OF BYTES TO BE TRANSFERRED
        .BTLPT==:5                      ;BYTE POINTER TO DATA TO BE LOADED
.BTDMP==:3                              ;DUMP MEMORY
        .BTDPT==:5                      ;BYTE POINTER TO DESTINATION OF DUMPED DATA
.BTIPR==:4                              ;INITIALIZE COMMUNICATIONS PROTOCOL
        .BTPRV==:1                      ;PROTOCOL VERSION NUMBER
.BTTPR==:5                              ;TERMINATE COMMUNICATIONS PROTOCOL
.BTSTS==:6                              ;RETURN PROTOCOL STATUS
        .BTCOD==:1                      ;STATUS CODE
.BTBEL==:7                              ;WAIT FOR DOORBELL
.BTRMP==:10                             ;READ MOP MESSAGE
        .BTMPT==:5                      ;POINTER TO DESTINATION FOR MOP MESSAGE


;CFORK

CR%MAP==:1B0                            ;SET MAP FOR NEW FORK TO POINT TO
                                        ; THIS PROCESS
CR%CAP==:1B1                            ;MAKE CAPABILITIES IDENTICAL
CR%ACS==:1B3                            ;SET ACS FROM BLOCK
CR%ST==:1B4                             ;START PROCESS AT PC
CR%PCV==:777777B35                      ;VALUE OF PC


;CHFDB

CF%NUD==:1B0                            ;NO UPDATE DIRECTORY
CF%DSP==:777B17                         ;FDB DISPLACEMENT
CF%JFN==:777777B35                      ;JFN
```

```
;CHKAC JSYS DEFINITIONS

;CHKAC FLAG DEFINITIONS

CK%JFN==:1B0                    ;JFN IS GIVEN AS AN ARGUMENT

;CHKAC ARGUMENT BLOCK OFFSET VALUES

.CKAAC==:0                      ;ACCESS CODE
.CKALD==:1                      ;LOGGED IN USER NUMBER OF USER
.CKACD==:2                      ;CONNECTED DIR NUMBER OF USER
.CKAEC==:3                      ;ENABLED CAPABILITIES OF USER BEING CHK'D
.CKAUD==:4                      ;DIR NUMBER OF DIRECTORY CONTAINING FILE
.CKAPR==:5                      ;PROTECTION OF FILE

;CHKAC ACCESS CODES

.CKARD==:0                      ;READ AN EXISTING FILE
.CKAWT==:1                      ;WRITE AN EXISTING FILE
.CKAWR==:1                      ; (ANOTHER NAME FOR ABOVE)
.CKAEX==:2                      ;EXECUTE AN EXISTING FILE
.CKAAP==:3                      ;APPEND TO AN EXISTING FILE
.CKADL==:4                      ;GET DIR LISTING OF AN EXISTING FILE
.CKADR==:6                      ;READ THE DIRECTORY
.CKAOF==:7                      ;OPEN FILES IN DIR (NOT IMPLEMENTED)
.CKACN==:10                     ;CONNECT TO A DIR
.CKACF==:11                     ;CREATE FILES IN DIR


;CLOSF

CO%NRJ==:1B0                    ;NO RELEASE JFN
CO%WCL==:1B1          ;TOPS20AN ;WAIT UNTIL MATCHING CLS IS RECEIVED
CO%JFN==:777777B35              ;JFN


;CLZFF

CZ%NIF==:1B0                    ;NO INFERIOR FORK FILES
CZ%NSF==:1B1                    ;NO SELF FORK FILES
CZ%NRJ==:1B2                    ;NO RELEASE JFN
CZ%NCL==:1B3                    ;NO CLOSE FILE
CZ%UNR==:1B4                    ;UNRESTRICT
CZ%ARJ==:1B5                    ;ALWAYS RELEASE JFN
CZ%ABT==:1B6                    ;ABORT
CZ%NUD==:1B7                    ;NO UPDATE DIRECTORY
CZ%PRH==:777777B35              ;PROCESS HANDLE
```

```
;CNDIR

CN%CKP==:1B0                    ;CHECK PASSWORD ONLY
CN%NOC==:1B1                    ;NO CONNECT
CN%JOB==:1B2                    ;DOING CONNECT FOR ANOTHER JOB
CN%DIR==:777777B35              ;DIRECTORY NUMBER


;COMND

;COMND - COMMAND STATE BLOCK

.CMFLG==:0                      ;USER FLAGS,,REPARSE DISPATCH ADDRESS
.CMIOJ==:1                      ;INJFN,,OUTJFN
.CMRTY==:2                      ;^R BUFFER POINTER
.CMBFP==:3                      ;PTR TO TOP OF BUFFER
.CMPTR==:4                      ;PTR TO NEXT INPUT TO BE PARSED
.CMCNT==:5                      ;COUNT OF SPACE LEFT IN BUFFER AFTER PTR
.CMINC==:6                      ;COUNT OF CHARACTERS FOLLOWING PTR
.CMABP==:7                      ;ATOM BUFFER POINTER
.CMABC==:10                     ;ATOM BUFFER SIZE
.CMGJB==:11                     ;ADR OF GTJFN ARG BLOCK
 CM%GJB==:777777                ;ADR OF GTJFN ARG BLOCK

;COMND - FUNCTION DESCRIPTOR BLOCK

.CMFNP==:0                      ;FUNCTION AND POINTER
  CM%FNC==:777B8                ;FUNCTION CODE
  CM%FFL==:777B17               ;FUNCTION-SPECIFIC FLAGS
  CM%LST==:777777               ;LIST POINTER TO OTHER BLOCKS
.CMDAT==:1                      ;DATA FOR FUNCTION
.CMHLP==:2                      ;HELP TEXT POINTER
.CMDEF==:3                      ;DEFAULT STRING POINTER


;COMND - MACRO FOR BUILDING FUNCTION DESCRIPTOR BLOCK

DEFINE FLDDB. (TYP,FLGS,DATA,HLPM,DEFM,LST)<
        ..XX==<FLD(TYP,CM%FNC)>+FLGS+<Z LST>
   IFNB <HLPM>,<..XX=CM%HPP!..XX>
   IFNB <DEFM>,<..XX=CM%DPP!..XX>
        ..XX
   IFNB <DATA>,<DATA>
   IFB <DATA>,<0>
   IFNB <HLPM>,<POINT 7,[ASCIZ \HLPM\]>
   IFB <HLPM>,<IFNB <DEFM>,<0>>
   IFNB <DEFM>,<POINT 7,[ASCIZ \DEFM\]>>
```

```
;COMND - FLAGS IN .CMFLG

CM%ESC==:1B0                    ;ESC SEEN
CM%NOP==:1B1                    ;NO PARSE
CM%EOC==:1B2                    ;END OF COMMAND SEEN
CM%RPT==:1B3                    ;REPEAT PARSE NEEDED
CM%SWT==:1B4                    ;SWITCH TERMINATED WITH ":"
CM%PFE==:1B5                    ;PREVIOUS FIELD ENDED WITH ESC
CM%RAI==:1B6                    ;RAISE INPUT
CM%XIF==:1B7                    ;EXCLUDE INDIRECT FILES
CM%WKF==:1B8                    ;WAKEUP AFTER EACH FIELD

;FUNCTION BLOCK FLAGS (IN WORD .CMFNP)

CM%PO==:1B14                    ;PARSE-ONLY
CM%HPP==:1B15                   ;HELP POINTER PRESENT
CM%DPP==:1B16                   ;DEFAULT POINTER PRESENT
CM%SDH==:1B17                   ;SUPPRESS DEFAULT HELP MESSAGE

;FLAGS FOR CMDIR FUNCTION

CM%DWC==:1B0                    ;DIRECTORY WILD CARDING ALLOWED

;FLAGS FOR CMTAD FUNCTION

CM%IDA==:1B0                    ;INPUT DATE
CM%ITM==:1B1                    ;INPUT TIME
CM%NCI==:1B2                    ;NO CONVERT TO INTERNAL

;FLAGS IN KEYWORD TABLE (FIRST WORD OF STRING IF B0-6 = 0)

CM%INV==:1B35                   ;INVISIBLE
CM%NOR==:1B34                   ;NO-RECOGNIZE (PLACEHOLDER)
CM%ABR==:1B33                   ;ABBREVIATION FOR ANOTHER ENTRY
CM%FW==:1B7                     ;FLAG WORD (MUST ALWAYS BE ON)
```

```
;COMND - FUNCTION CODES

.CMKEY==:0                      ;KEYWORD
.CMNUM==:1                      ;NUMBER
.CMNOI==:2                      ;NOISE WORD
.CMSWI==:3                      ;SWITCH
.CMIFI==:4                      ;INPUT FILE
.CMOFI==:5                      ;OUTPUT FILE
.CMFIL==:6                      ;GENERAL FILESPEC
.CMFLD==:7                      ;ARBITRARY FIELD
.CMCFM==:10                     ;CONFIRM
.CMDIR==:11                     ;DIRECTORY NAME
.CMUSR==:12                     ;USER NAME
.CMCMA==:13                     ;COMMA
.CMINI==:14                     ;INIT LINE
.CMFLT==:15                     ;FLOATING POINT NUMBER
.CMDEV==:16                     ;DEVICE NAME
.CMTXT==:17                     ;TEXT TO ACTION CHAR
.CMTAD==:20                     ;TIME AND DATE
.CMQST==:21                     ;QUOTED STRING
.CMUQS==:22                     ;UNQUOTED STRING
.CMTOK==:23                     ;TOKEN
.CMNUX==:24                     ;NUMBER DELIMITED BY NON-DIGIT
.CMACT==:25                     ;ACCOUNT
.CMNOD==:26                     ;NODE NAME
```

```
;CRDIR

CD%LEN==:1B0                        ;FLAGS ,, LENGTH OF CRDIR BLOCK
CD%PSW==:1B1                        ;SET PASSWORD STRING
CD%LIQ==:1B2                        ;SET LOGGED IN QUOTA
CD%PRV==:1B3                        ;SET PRIVILEGES
CD%MOD==:1B4                        ;SET MODE BITS
CD%LOQ==:1B5                        ;SET LOGGED OUT QUOTA
CD%NUM==:1B6                        ;SET DIRECTORY NUMBER FROM PARAM BLK
CD%FPT==:1B7                        ;SET DEFAULT FILE PROTECTION
CD%DPT==:1B8                        ;SET DIRECTORY PROTECTION
CD%RET==:1B9                        ;SET DEFAULT RETENTION COUNT
CD%LLD==:1B10                       ;SET LAST LOGIN DATE
CD%UGP==:1B11                       ;SET USER GROUPS
CD%DGP==:1B12                       ;SET DIRECTORY GROUPS
CD%SDQ==:1B13                       ;SET SUBDIRECTORY QUOTA
CD%CUG==:1B14                       ;SET CREATABLE USER GROUPS
CD%DAC==:1B15                       ;SET DEFAULT ACCOUNT
CD%DEL==:1B17                       ;DELETE DIRECTORY
CD%APB==:777777B35                  ;ADDRESS OF PARAMETER BLOCK
.CDLEN==:0                          ;LENGTH OF ARGUMENT BLOCK
        CD%NSQ==:1B0                ;DO NOT UPDATE QUOTAS OF SUPERIOR DIR
        CD%NCE==:1B1                ;DO NOT CHANGE PARAMETERS OF EXISTING DIRS
.CDPSW==:1                          ;POINTER TO PASSWORD STRING
.CDLIQ==:2                          ;LOGGED IN QUOTA
.CDPRV==:3                          ;PRIVILEGE WORD
.CDMOD==:4                          ;MODE WORD
        CD%DIR==:1B0                ;DIRECTORY NAME FOR CNDIR ONLY (FILES ONLY)
        CD%ANA==:1B1                ;ALPHANUMERIC ACCOUNTS
        CD%RLM==:1B2                ;REPEAT LOGIN MESSAGES
.CDLOQ==:5                          ;LOGGED OUT QUOTA
.CDNUM==:6                          ;DIRECTORY NUMBER
.CDFPT==:7                          ;DEFAULT FILE PROTECTION
.CDDPT==:10                         ;DIRECTORY PROTECTION
.CDRET==:11                         ;DEFAULT RETENTION COUNT
.CDLLD==:12                         ;LAST LOGIN DATE
.CDUGP==:13                         ;USER GROUPS
.CDDGP==:14                         ;DIRECTORY GROUPS
.CDSDQ==:15                         ;MAXIMUM NUMBER OF SUBDIRECTORIES
.CDCUG==:16                         ;POINTER TO CREATABLE USER GROUP LIST
.CDDAC==:17                         ;POINTER TO DEFAULT ACCOUNT
```

```
;CRJOB

CJ%LOG==:1B0                        ;ATTEMPT TO LOG IN THE NEW JOB
CJ%NAM==:1B1                        ;USE NAME AND PSWD IN ARG BLK
CJ%ACT==:3B3                        ;WHERE TO GET ACCOUNT
        .CJUCA==:0                  ;USE CURRENT ACCT OF CREATOR
        .CJUAA==:1                  ;USE ACCOUNT IN ARG BLOCK
        .CJUDA==:2                  ;USE DEFAULT ACCOUNT OF NEW USER
CJ%ETF==:1B4                        ;PUT EXEC IN TOP FORK
CJ%FIL==:1B5                        ;GET FILE IN ARG BLOCK
CJ%ACS==:1B6                        ;LOAD THE ACS FROM ARG BLOCK
CJ%OWN==:1B7                        ;RETAIN OWNERSHIP OF NEW JOB
CJ%WTA==:1B8                        ;NEW JOB WAITS TIL ATTACHED
CJ%NPW==:1B9                        ;NO PASSWORD CHECK AT LOGIN TIME
CJ%NUD==:1B10                       ;NO UPDATE OF LAST-LOGIN DATE
CJ%SPJ==:1B11                       ;DO SPJFN IN NEW JOB FROM ARG BLK
CJ%CAP==:1B12                       ;PASS ENABLED CAPABILITIES AS ALLOWED
CJ%CAM==:1B13                       ;CAPABILITY MASK AT LOGIN
CJ%SLO==:1B14                       ;SIGNAL (IPCF) AT LOGOUT TIME
CJ%DSN==:1B17                       ;DISOWN EXISTING JOB # IN 3

.CJNAM==:0                          ;NAME STRING POINTER
.CJPSW==:1                          ;PASSWORD STRING POINTER
.CJACT==:2                          ;ACCOUNT DESIGNATOR/STRING
.CJFIL==:3                          ;FILE NAME STRING POINTER
.CJSFV==:4                          ;SFRKV OFFSET
.CJTTY==:5                          ;TTY DESIGNATOR, OR NULL DESIGNATOR
.CJTIM==:6                          ;TIME LIMIT
.CJACS==:7                          ;ADDRESS OF 16. WORDS OF AC'S
.CJEXF==:10                         ;EXEC FLAGS, FOR EXEC AC1
.CJPRI==:11                         ;PRIMARY JFN'S FOR SPJFN IN NEW JOB
.CJCPU==:12                         ;CPU LIMIT (0 IF NONE)
.CJCAM==:13                         ;CAPABILITY MASK TO APPLY TO LOGIN
.CJSLO==:14                         ;PID TO SIGNAL AT LOGOUT TIME

CR%PRA==:2545                       ;MAGIC # FOR EXEC/CRJOB LINKAGE VIA PRARG


;CRLNM

.CLNJ1==:0                          ;DELETE 1 LOGICAL NAME FROM JOB
.CLNS1==:1                          ;DELETE 1 LOGICAL NAME FROM SYSTEM
.CLNJA==:2                          ;DELETE ALL JOB WIDE LOGICAL NAMES
.CLNSA==:3                          ;DELETE ALL SYSTEM LOGICAL NAMES
.CLNJB==:4                          ;CREATE A JOB WIDE LOGICAL NAME
.CLNSY==:5                          ;CREATE A SYSTEM WIDE LOGICAL NAME
```

```
;DELDF

DD%DTF==:1B0                    ;DELETE TEMPORARY FILES
DD%DNF==:1B1                    ;DELETE NONEXISTENT FILES
DD%RST==:1B2                    ;REBUILD THE SYMBOL TABLE
DD%CHK==:1B3                    ;CHECK THE DIR FOR CONSISTENCY ONLY


;DELF

DF%NRJ==:1B0                    ;DON'T RELEASE JFN
DF%EXP==:1B1                    ;EXPUNGE CONTENTS
DF%FGT==:1B2                    ;FORGET (EXPUNGE W/O DEASSIGNING ADDRESSES)
DF%DIR==:1B3                    ;DELETE, FORGET, AND EXPUNGE A DIRECTORY
                                ; FILE.  (ONLY IF ^E-CREATE KILL FAILED)
DF%JFN==777777B35               ;JFN


;DIAG JSYS DEFINITIONS

DG%ADT==:7B2                    ;ADDRESS TYPE FIELD
DG%DVC==:177B9                  ;DEVICE CODE FIELD
        .DGRH0==:130            ;MBC0
        .DGRH7==:137            ;MBC7
DG%UNI==:77B29                  ;UNIT NUMBER
DG%SUN==:77B35                  ;SUBUNIT NUMBER

;DIAG JSYS FUNCTION CODES

.DGACU==:1                      ;ASSIGN DEVICE
.DGACH==:2                      ;ASSIGN CONTROLLER AND ALL DEVICES
.DGRCH==:3                      ;RELEASE DEVICE(S)
.DGSCP==:4                      ;SETUP CHANNEL PROGRAM
.DGRCP==:5                      ;RELEASE CHANNEL PROGRAM
.DGGCS==:6                      ;GET CHANNEL STATUS


;DSKAS

DA%DEA==:1B0                    ;DEASSIGN DISK ADDRESS
DA%ASF==:1B1                    ;ASSIGN FREE PAGE
DA%CNV==:1B2                    ;CONVERT SOFTWARE TO HARDWARE ADDRESS
DA%HWA==:1B3                    ;HARDWARE ADDRESS GIVEN
DA%INI==:1B4                    ;INITIALIZE THE BIT TABLE
DA%WRT==:1B5                    ;WRITE THE BIT TABLE FILE
DA%ADR==:777777B35              ;DISK ADDRESS
```

```
;DVCHR AND DVCH1 BIT DEFINITIONS

DV%OUT==:1B0                    ;DEVICE CAN DO OUTPUT
DV%IN==:1B1                     ;DEVICE CAN DO INPUT
DV%DIR==:1B2                    ;DEVICE HAS A DIRECTORY
DV%AS==:1B3                     ;DEVICE IS ASSIGNABLE
DV%MDD==:1B4                    ;DEVICE IS A MULTIPLE DIRECTORY DEVICE
DV%AV==:1B5                     ;DEVICE IS AVAILABLE TO THIS JOB
DV%ASN==:1B6                    ;DEVICE IS ASSIGNED BY ASND
DV%MDV==:1B7                    ;RESERVED (HISTORICAL)
DV%MNT==:1B8                    ;DEVICE IS MOUNTED
DV%TYP==:777B17                 ;DEVICE TYPE FIELD
DV%MOD==:177777B35              ;DEVICE DATA MODE
DV%M0==:1B35                    ;DEVICE CAN BE OPENED IN MODE 0
DV%M1==:1B34                    ;DEVICE CAN BE OPENED IN MODE 1
DV%M2==:1B33                    ;DEVICE CAN BE OPENED IN MODE 2
DV%M3==:1B32                    ;DEVICE CAN BE OPENED IN MODE 3
DV%M4==:1B31                    ;DEVICE CAN BE OPENED IN MODE 4
DV%M5==:1B30                    ;DEVICE CAN BE OPENED IN MODE 5
DV%M6==:1B29                    ;DEVICE CAN BE OPENED IN MODE 6
DV%M7==:1B28                    ;DEVICE CAN BE OPENED IN MODE 7
DV%M10==:1B27                   ;DEVICE CAN BE OPENED IN MODE 10
DV%M11==:1B26                   ;DEVICE CAN BE OPENED IN MODE 11
DV%M12==:1B25                   ;DEVICE CAN BE OPENED IN MODE 12
DV%M13==:1B24                   ;DEVICE CAN BE OPENED IN MODE 13
DV%M14==:1B23                   ;DEVICE CAN BE OPENED IN MODE 14
DV%M15==:1B22                   ;DEVICE CAN BE OPENED IN MODE 15
DV%M16==:1B21                   ;DEVICE CAN BE OPENED IN MODE 16
DV%M17==:1B20                   ;DEVICE CAN BE OPENED IN MODE 17
D1%SPL==:1B0                    ;DEVICE IS SPOOLED
D1%ALC==:1B1                    ;DEVICE IS UNDER CONTROL OF ALLOCATOR
D1%VVL==:1B2                    ;VOLUME VALID
D1%NIU==:1B3                    ;DEVICE SLOT IS NOT IN USE (FOR STRUCTURES
                                ; NOT YET MOUNTED)
D1%INI==:1B4                    ;DEVICE IS BEING INITIALIZED (STRUCTURE
                                ; IS AVAILABLE ONLY TO THE FORK WHOSE NUMBER
                                ; IS STORED IN SDBSTS)
```

```
;DEVICE TYPE DEFINITIONS

.DVDSK==:0                          ;DISK
.DVMTA==:2                          ;MAGTAPE
.DVDTA==:3                          ;DECTAPE
.DVPTR==:4                          ;PAPER TAPE READER
.DVPTP==:5                          ;PAPER TAPE PUNCH
.DVDSP==:6                          ;DISPLAY
.DVLPT==:7                          ;LINE PRINTER
.DVCDR==:10                         ;CARD READER
.DVFE==:11                          ;FRONT END DEVICE
.DVTTY==:12                         ;TERMINAL
.DVPTY==:13                         ;PTY
.DVNUL==:15                         ;NULL DEVICE
.DVNET==:16                         ;ARPA NETWORK
.DVPLT==:17                         ;PLOTTER
.DVDCN==:22                         ;DECNET ACTIVE COMPONENT
.DVSRV==:23                         ;DECENT PASSIVE COMPONENT


;DSKOP

DOP%SA==:1B0                        ;SOFTWARE ADDRESS
DOP%AT==:3B1                        ;ADDRESS TYPE FIELD
 .DOPPU==:1                         ;PHYSICAL CHANNEL AND UNIT
  DOP%CN==:37B6                     ;CHANNEL NUMBER
  DOP%UN==:77B12                    ;UNIT NUMBER
  DOP%UA==:37777777                 ;UNIT ADDRESS
 .DOPSR==:2                         ;STRUCTURE AND RELATIVE ADDRESS
  DOP%SN==:777B10                   ;STRUCTURE NUMBER
  DOP%RA==:177777777                ;RELATIVE ADDRESS
DOP%EO==:1B10                       ;ERROR IF UNIT OFFLINE
DOP%IL==:1B11                       ;INHIBIT ERROR LOGGING
DOP%IR==:1B12                       ;INHIBIT ERROR RECOVERY
DOP%WR==:1B14                       ;WRITE
DOP%CT==:777777B35                  ;WORD COUNT


;DUMPI/DUMPO

DM%NWT==:1B0                        ;NO WAIT FOR COMPLETION
DM%FIN==:1B1                        ;FINISH PREVIOUS REQUEST
                                    ;***NOT IMPLEMENTED YET***
DM%PTR==:777777B35                  ;POINTER TO COMMAND LIST
```

```
;DEFINE DECNET DISCONNECT CODES. THESE ARE STIPULATED BY THE NSP SPEC
;AND MAY HAVE MEANININGS NOT IMPLIED BY THE COMMENTS

.DCX0==:0                       ;NO SPECIAL ERROR
.DCX1==:1                       ;RESOURCE ALLOCATION FAILURE
.DCX2==:2                       ;DESTINATION NODE DOES NOT EXIST
.DCX3==:3                       ;NODE SHUTTING DOWN
.DCX4==:4                       ;DESTINATION PROCESS DOES NOT EXIST
.DCX5==:5                       ;INVALID NAME FIELD
.DCX11==:^D11                   ;USER ABORT (ASYNCHRONOUS DISCONNECT)
.DCX21==:^D21                   ;CI WITH ILLEGAL DESTINATION ADDRESS
.DCX22==:^D22                   ;CC WITH ILLEGAL DESTINATION ADDRESS
.DCX23==:^D23                   ;CI OR CC WITH ZERO SOURCE ADDRESS
.DCX32==:^D32                   ;TOO MANY CONNECTIONS TO NDOE
.DCX33==:^D33                   ;TOO MANY CONNECTIONS TO DEST. PROCESS
.DCX34==:^D34                   ;ACCESS NOT PERMITTED
.DCX35==:^D35                   ;LOGICAL LINK SERVICES MISMATCH
.DCX36==:^D36                   ;INVALID ACCOUNT
.DCX37==:^D37                   ;SEGSIZE TOO SMALL
.DCX38==:^D38                   ;PROCESS ABORTED
.DCX39==:^D39                   ;NO PATH TO DESTINATION NODE
.DCX40==:^D40                   ;LINK ABORTED DUE TO DATA LOSS
.DCX41==:^D41                   ;DESTINATION PROCESS DOES NOT EXIST
.DCX42==:^D42                   ;CONFIRMATION IF DI
.DCX43==:^D43                   ;IMAGE DATA FIELD TOO LONG


;EFACT - FACT FILE ENTRY DEFINITIONS

.EFHDR==:0                      ;HEADER WORD
        EF%COD==:777B8          ;ENTRY TYPE CODE
        EF%JOB==:777B17         ;JOB NUMBER
        EF%LIN==:7777B29        ;LINE NUMBER
        EF%SIZ==:77B35          ;ENTRY SIZE
.EFUSR==:1                      ;USER NUMBER WORD
.EFTAD==:2                      ;TIME AND DATE OF ENTRY

; FACT FILE ENTRY TYPE CODES

.EFLGI==:501                    ;LOGIN
.EFLGO==:141                    ;LOGOUT
.EFCAC==:502                    ;CHANGE ACCOUNT
.EFATT==:142                    ;CONSOLE ATTACH
.EFDET==:143                    ;CONSOLE DETACH
.EFCHK==:201                    ;CHECKPOINT
.EFSDU==:540                    ;START DISK-UTILIZATION ENTRIES
.EFDSK==:601                    ;DISK SPACE UTILIZATION
.EFTIM==:741                    ;TIME SET
.EFRES==:740                    ;SYSTEM RESTARTED
.EFLPT==:401                    ;LINE PRINTER USAGE
.EFCDR==:402                    ;CARD READER USAGE
```

```
;ENQ/DEQ BIT DEFINITIONS AND FUNCTION CODES

;FUNCTION CODES

.ENQBL==:0                      ;ENQ BLOCK OPTION
.ENQAA==:1                      ;ENQ ALLOCATE ONLY IF AVAILABLE
.ENQSI==:2                      ;ENQ SOFTWARE INTERRUPT WHEN LOCKED
.ENQMA==:3                      ;ENQ MODIFY ACCESS
.DEQDR==:0                      ;DEQ RESOURCE
.DEQDA==:1                      ;DEQ ALL RESOURCES OF THIS FORK
.DEQID==:2                      ;DEQ THIS ID NUMBER
.ENQCS==:0                      ;ENQC STATUS
.ENQCG==:1                      ;ENQC GET ENQ/DEQ QUOTA FOR A JOB
.ENQCC==:2                      ;ENQC CHANGE ENQ/DEQ QUOTA FOR A JOB
.ENQCD==:3                      ;ENQC DUMP LOCKS AND QUEUE ENTRIES


;BIT DEFINITIONS

EN%SHR==:1B0                    ;SHARABLE REQUEST
EN%BLN==:1B1                    ;BYPASS LEVEL NUMBER
EN%NST==:1B2                    ;ALLOW NESTING
EN%LTL==:1B3                    ;LONG TERM LOCK
EN%LVL==:777B17                 ;LEVEL NUMBER
EN%JOB==:777777B35              ;JOB NUMBER
EN%QCE==:1B0                    ;ERROR CODE IN RH OF STATUS WORD
EN%QCL==:1B0                    ;LOCK DUMP (.ENQCD ONLY)
EN%QCO==:1B1                    ;THIS FORK OWNS THE LOCK
EN%QCQ==:1B2                    ;THIS FORK IS IN THE QUEUE FOR THIS LOCK
EN%QCT==:1B2                    ;LOCK CONTAINS A TEXT STRING
EN%QCX==:1B3                    ;THE LOCK IS LOCKED EXCLUSIVELY
EN%QCB==:1B4                    ;USER IS BLOCKED FOR LOCK

;ENQ/DEQ ARGUMENT BLOCK DATA STRUCTURE

.ENQLN==:0                      ;# OF LOCKS ,, LENGTH OF ARGUMENT BLOCK
        .ENHLN==:77B5           ;LENGTH OF HEADER AREA
        .ENNLK==:7777B17        ;NUMBER OF LOCKS
        .ENALN==:777777B35      ;LENGTH OF ARGUMENT BLOCK
.ENQID==:1                      ;PSI CHANNEL # ,, REQUEST ID
.ENQLV==:2                      ;FLAGS & LEVEL NUMBER ,, JFN, -1, -2, OR -3
.ENQUC==:3                      ;STRING POINTER OR USER CODE
.ENQRS==:4                      ;# OF RESOURCES IN POOL ,, # OF RESOURCES WANTED
.ENQMS==:5                      ;ADDRESS OF RESOURCE BLOCK
```

```
;ENQC DUMP DATA STRUCTURE

.ENQDF==:0                     ;FLAGS + LEVEL # ,, OFN, 400000+JOB #, -2, OR -3
.ENQDR==:1                     ;TOTAL RESOURCES IN POOL ,, RESOURCES REMAINING
.ENQDT==:2                     ;TIME STAMP OF LAST REQUEST LOCKED
.ENQDC==:3                     ;USER CODE OF LOCK OR START OF TEXT STRING
.ENQDF==:0                     ;FLAGS + PSI # ,, JOB # OF Q-ENTRY CREATOR
.ENQDI==:1                     ;GROUP # OR # REQUESTED ,, ENQ ID


;FLOUT/DFOUT
;FORMAT CONTROL WORD

FL%SGN==:3B1                   ;FIRST FIELD SIGN CONTROL
 .FLDIG==:0                    ;DIGIT
 .FLSPC==:1                    ;SPACE
 .FLPLS==:2                    ;PLUS SIGN
 .FLSPA==:3                    ;SPACE
FL%JUS==;3B3                   ;FIRST FIELD JUSTIFICATION CONTROL
 .FLLSP==:0                    ;LEADING SPACES
 .FLLZR==:1                    ;LEADING ZEROS
 .FLLAS==:2                    ;LEADING ASTERISKS
 .FLTSP==:3                    ;TRAILING SPACES
FL%ONE==:1B4                   ;FIRST FIELD NONBLANK
FL%DOL==:1B5                   ;DOLLAR SIGN PREFIX
FL%PNT==:1B6                   ;DECIMAL POINT
FL%EXP==:3B8                   ;THIRD FIELD EXPONENT CONTROL
 .FLEXN==:0                    ;NO EXPONENT
 .FLEXE==:1                    ;E EXPONENT PREFIX
 .FLEXD==:2                    ;D EXPONENT PREFIX
 .FLEXM==:3                    ;*10^ EXPONENT PREFIX
FL%ESG==:3B10                  ;EXPONENT SIGN CONTROL
 .FLDGE==:0                    ;DIGIT
 .FLPLE==:1                    ;PLUS SIGN
 .FLSPE==:2                    ;SPACE
 .FLDGT==:3                    ;DIGIT
FL%OVL==:1B11                  ;COLUMN OVERFLOW
FL%RND==:37B17                 ;DIGIT POSITION FOR ROUNDING
FL%FST==:77B23                 ;FIRST FIELD WIDTH
FL%SND==:77B29                 ;SECOND FIELD WIDTH
FL%THD==:77B35                 ;THIRD FIELD WIDTH
```

```
;GDSTS

;SEE MTOPR FOR CARD READER AND LINE PRINTER STATUS BITS
;SEE GENERAL FIELD AND VALUE SECTION FOR MAGTAPE STATUS BITS
;SEE TOPS20AN SECTION FOR NETWORK STATUS BITS

.GDFSM==:17B3           ;TOPS20AN ;FINITE MACHINE STATE


;GET

GT%ADR==:1B19                    ;USE ADDRESS LIMITS IN AC2
GT%PRL==:1B20                    ;PRELOAD PAGES
GT%NOV==:1B21                    ;DON'T OVERLAY EXISTING PAGES
GT%FL2==:1B22                    ;IF ON, AC3 CONTAINS FLAGS
```

```
;GETAB - TABLE INDICES

.JOBTT==:0                        ;JOB NUMBER TO TTY NUMBER
.JOBRT==:1                        ;JOB RUNTIME
.TICKP==:2                        ;TICKS PER SECOND
.JOBDI==:3                        ;JOB NUMBER TO DIRECTORY NUMBERS (OBS)
.TTYJO==:4                        ;TTY NUMBER TO JOB NUMBER
.NCPGS==:5                        ;NUMBER PHYSICAL CORE PAGES
.DEVNA==:6                        ;DEVICE NAME
.DEVCH==:7                        ;DEVICE CHARACTERISTICS
.DEVUN==:10                       ;DEVICE UNIT NUMBERS
.DSKER==:11                       ;DISK ERROR WORDS
.DRMER==:12                       ;DRUM ERROR WORDS
.SYSVE==:13                       ;VERSION TEXT
.SYSTA==:14                       ;STATISTICS
.QTIME==:15                       ;SCHED QUEUE TIMES
.JOBNA==:16                       ;JOB NUMBER TO PROGRAM NAME
.SNAME==:17                       ;SUBSYSTEM NAME
.STIME==:20                       ; " TIME
.SPFLT==:21                       ; " PAGE FAULTS
.SSIZE==:22                       ; " SIZE INTEGRAL
.SNBLK==:23                       ; " NUMBER WAKEUPS
.DBUGS==:24                       ;DBUGSW, DCHKSW
.LOGDE==:25                       ;LOG, JOB 0 DESIGNATORS
.PTYPA==:26                       ;PTY PARAMETERS
.SYMTA==:27                       ;GTTAB SYMBOL TABLE
.DWNTI==:30                       ;HSYS VARIABLES
.JOBPN==:31                       ;JOB NUMBER TO PROGRAM NAME
.BLDTD==:32                       ;MONITOR BUILD TIME AND DATE
.LSTDR==:33                       ;LAST DIR NUMBER ASSIGNED (OBS)
.APRID==:34                       ;APR SERIAL NUMBER
.HQLAV==:35                       ;HIGH QUEUE LOAD AVERAGES
.LQLAV==:36                       ;LOW QUEUE LOAD AVERAGES
.NETRD==:37             ;TOPS20AN ;ARPANET STATUS
.IMPHR==:40             ;TOPS20AN ;HOST READY
.HSTST==:41             ;TOPS20AN ;DEAD HOST STATUS
.HSTNA==:42             ;TOPS20AN ;HOST NAMES
.HOSTN==:43             ;TOPS20AN ;HOST NAME INDEX
.NETLS==:44             ;TOPS20AN ;LOCAL SOCKET
.NETFS==:45             ;TOPS20AN ;FOREIGN SOCKET
.NETAW==:46             ;TOPS20AN ;ARPA CONNECTION ADDRESS
.NETBA==:47             ;TOPS20AN ;BIT ALLOCATION
.NETST==:50             ;TOPS20AN ;CONNECTION STATUS
.NETBU==:51             ;TOPS20AN ;ARPANET BUFFERS
.NETBT==:52             ;TOPS20AN ;BYTE COUNT STATISTICS
.IMPL1==:53             ;TOPS20AN ;IMP LINK TABLE ONE
.IMPL2==:54             ;TOPS20AN ;IMP LINK TABLE TWO
.IMPL3==:55             ;TOPS20AN ;IMP LINK TABLE THREE
.IMPL4==:56             ;TOPS20AN ;IMP LINK TABLE FOUR
.LHOST==:57             ;TOPS20AN ;LOCAL HOST NUMBER
.JBONT==:60                       ;OWNING JOB
```

```
;GETJI

.JIJNO==:0                      ;JOB NUMBER
.JITNO==:1                      ;TERMINAL NUMBER
.JIUNO==:2                      ;USER NUMBER
.JIDNO==:3                      ;DIRECTORY NUMBER
.JISNM==:4                      ;SUBSYS NAME
.JIPNM==:5                      ;PROGRAM NAME
.JIRT==:6                       ;RUN TIME
.JICPJ==:7                      ;CONTROLLING PTY JOB NUMBER
.JIRTL==:10                     ;RUN TIME LIMIT (SET BY TIMER JSYS)
.JIBAT==:11                     ;CONTROLLED BY BATCH
.JIDEN==:12                     ;MAGTAPE DEFAULT DENSITY
.JIPAR==:13                     ;MAGTAPE DEFAULT PARITY
.JIDM==:14                      ;MAGTAPE DEFAULT DATA MODE
.JIRS==:15                      ;MAGTAPE DEFAULT RECORD SIZE
.JIDFS==:16                     ;DEFERRED SPOOLING
.JILNO==:17                     ;LOGGED-IN DIRECTORY NUMBER
.JISRM==:20                     ;POINTER TO JOB SESSION REMARK
.JILLN==:21                     ;LAST LOGIN DATE & TIME


;GFRKS

GF%GFH==:1B0                    ;GET RELATIVE FORK HANDLES
GF%GFS==:1B1                    ;GET FORK STATUS


;GFUST

.GFAUT==:0                      ;GET FILE AUTHOR
.GFLWR==:1                      ;GET FILE LAST WRITER
```

```
;GTJFN DEFINITIONS

;FLAGS PROVIDED TO GTJFN ON CALL

GJ%FOU==:1B0                    ;FILE IS FOR OUTPUT USE
GJ%NEW==:1B1                    ;NEW FILE ONLY
GJ%OLD==:1B2                    ;OLD FILE ONLY
GJ%MSG==:1B3                    ;PRINT AN APPROPRIATE MESSAGE
GJ%CFM==:1B4                    ;CONFIRMATION IS REQUIRED
GJ%TMP==:1B5                    ;TEMPORARY
GJ%NS==:1B6                     ;DONT SEARCH SEARCH LISTS
GJ%ACC==:1B7                    ;NO ACCESS BY OTHER FORKS
GJ%DEL==:1B8                    ;IGNORE "DELETED" BIT
GJ%JFN==:3B10                   ;JFN USE FIELD
.GJDNU==:0                      ;DO NOT USE JFN PROVIDED
.GJERR==:2                      ;ERROR IF CANNOT USE JFN PROVIDED
.GJALT==:3                      ;USE ALTERNATE IF CANNOT USE GIVEN JFN
GJ%IFG==:1B11                   ;ACCEPT INPUT FILE GROUP DESCRIPTORS
GJ%OFG==:1B12                   ;ACCEPT OUTPUT FILE GROUP DESCRIPTORS
GJ%FLG==:1B13                   ;RETURN FLAGS
GJ%PHY==:1B14                   ;PHYSICAL DEVICE ONLY
GJ%XTN==:1B15                   ;EXTENDED FORMAT (E+11 EXISTS)
GJ%FNS==:1B16                   ;ACCUMULATOR 2 CONTAINS JOB FILE NUMBERS
GJ%SHT==:1B17                   ;SHORT CALL FORMAT

;FLAGS PROVIDED TO GTJFN (IN SECOND FLAG WORD)

G1%RND==:1B0                    ;RETURN ON NULL(IN ALTERNATE FLAG WORD)
G1%RBF==:1B1                    ;^R BUFFER IS DISJOINT (OBSOLETE)
G1%NLN==:1B2                    ;NO LONG NAMES
G1%RCM==:1B3                    ;RETURN CONFIRM MESSAGE
G1%RIE==:1B4                    ;RETURN WHEN MAIN STRING IS EMPTY
```

```
;FLAGS RETURNED BY GTJFN

GJ%DEV==:1B0                    ;ASTERISK WAS GIVEN FOR DEVICE
GJ%UNT==:1B1                    ;ASTERISK WAS GIVEN FOR UNIT
GJ%DIR==:1B2                    ;ASTERISK WAS GIVEN FOR DIRECTORY
GJ%NAM==:1B3                    ;ASTERISK WAS GIVEN FOR NAME
GJ%EXT==:1B4                    ;ASTERISK WAS GIVEN FOR EXTENSION
GJ%VER==:1B5                    ;ASTERISK WAS GIVEN FOR GENERATION
GJ%UHV==:1B6                    ;USE HIGHEST GENERATION
GJ%NHV==:1B7                    ;USE NEXT HIGHER GENERATION
GJ%ULV==:1B8                    ;USE LOWEST GENERATION
GJ%PRO==:1B9                    ;PROTECTION GIVEN
GJ%ACT==:1B10                   ;ACCOUNT GIVEN
GJ%TFS==:1B11                   ;TEMPORARY FILE SPECIFIED (;T)
GJ%GND==:1B12                   ;COMPLEMENT OF GJ%DEL ON CALL

;GTJFN TABLE OFFSETS

.GJGEN==:0                      ;FLAGS ,, GENERATION
.GJDEF==:<Z 0>                  ;DEFAULT GENERATION
.GJNHG==:<Z -1>                 ;NEXT HIGHER GENERATION
.GJLEG==:<Z -2>                 ;LOWEST EXISTING GENERATION
.GJALL==:<Z -3>                 ;ALL GENERATIONS (I.E., ;*)
.GJSRC==:1                      ;SOURCE JFN ,, OUTPUT JFN
.GJDEV==:2                      ;DEFAULT DEVICE
.GJDIR==:3                      ;DEFAULT DIRECTORY
.GJNAM==:4                      ;DEFAULT NAME
.GJEXT==:5                      ;DEFAULT EXTENSION
.GJPRO==:6                      ;DEFAULT PROTECTION
.GJACT==:7                      ;DEFAULT ACCOUNT
.GJJFN==:10                     ;DESIRED JFN
.GJF2==:11                      ;SECOND GROUP FLAGS,,COUNT
.GJCPP==:12                     ;COPY BUFFER POINTER
.GJCPC==:13                     ;COPY BUFFER COUNT
.GJRTY==:14                     ;RETYPE (^R) POINTER
.GJBFP==:15                     ;TOP OF BUFFER POINTER
.GJATR==:16                     ;POINTER TO ARBITRARY ATTRIBUTE BLOCK

;GNJFN - FLAGS RETURNED

GN%STR==:1B13                   ;STRUCTURE CHANGED
GN%DIR==:1B14                   ;DIRECTORY CHANGED
GN%NAM==:1B15                   ;NAME CHANGED
GN%EXT==:1B16                   ;EXTENSION CHANGED
```

```
;GTRPW

PF%USR==:1B0                    ;PAGE FAIL WORD - USER MODE REFERENCE
PF%WRT==:1B5                    ; " - WRITE REFERENCE
TSW%RD==:1B14                   ;TRAP STATUS WORD - READ
TSW%WT==:1B15                   ; " - WRITE
TSW%WR==:1B15                   ; (ANOTHER NAME FOR ABOVE)
TSW%EX==:1B16                   ; " - EXECUTE
TSW%MN==:1B17                   ; " - MONITOR MODE REFERENCE


;GTSTS BITS RETURNED IN 2

GS%OPN==:1B0                    ;FILE IS OPEN
GS%RDF==:1B1                    ;IF OPEN, FILE IS OPEN FOR READ
GS%WRF==:1B2                    ;IF OPEN, FILE IS OPEN FOR WRITE
GS%XCF==:1B3                    ;IF OPEN, FILE IS OPEN FOR EXECUTE
GS%RND==:1B4                    ;OK TO RESET BYTE POINTER
                                ; (FILE IS NOT APPEND)
GS%APT==:1B5                    ;ACCESS PER PAGE TABLE
                                ; (NOT IMPLEMENTED -- OBSOLETE)
GS%CAL==:1B6                    ;OK TO CALL AS A PROCEDURE
                                ; (NOT IMPLEMENTED -- OBSOLETE)
GS%LNG==:1B7                    ;FILE IS LONG
GS%EOF==:1B8                    ;AT END OF FILE ON READ
GS%ERR==:1B9                    ;FILE MAY BE IN ERROR
GS%NAM==:1B10                   ;FILE HAS A NAME (JFN EXISTS)
GS%AST==:1B11                   ;ONE OR MORE FIELDS OF NAME
                                ; IS WILD
GS%ASG==:1B12                   ;JFN IS BEING ASSIGNED
GS%HLT==:1B13                   ;TERMINATE ON I/O ERROR
GS%FRK==:1B17                   ;FILE IS RESTRICTED TO SOME FORK
GS%MOD==:17B35                  ;DATA MODE
        .GSNRM==:0              ;NORMAL MODE
        .GSIMG==:10             ;IMAGE (BINARY) MODE
        .GSDMP==:17             ;DUMP MODE
```

```
;HPTIM

.HPELP==:0                          ;ELAPSED TIME
.HPRNT==:1                          ;RUN TIME


;IDCNV (ALSO IDTNC AND ODCNV)

IC%DSA==:1B0                        ;DAYLIGHT SAVINGS IF APPROPRIATE
IC%ADS==:1B1                        ;APPLY DAYLIGHT SAVINGS
IC%UTZ==:1B2                        ;USE TIME ZONE GIVEN
IC%JUD==:1B3                        ;USE JULIAN DATE CONVERSION
IC%TMZ==:77B17                      ;TIME ZONE
IC%TIM==777777B35                   ;LOCAL TIME


;IDTIM & IDTNC

IT%NDA==:1B0                        ;NO DATE
IT%NNM==:1B1                        ;NO NUMERIC MONTH
IT%SNM==:1B2                        ;SECOND NUMBER IS MONTH
IT%ERR==:1B3                        ;ERROR IF NUMBERS ARE NOT IN SPECIFIED
                                    ; ORDER
IT%NTI==:1B6                        ;NO TIME
IT%NIS==:1B7                        ;NO SECONDS
IT%AIS==:1B8                        ;ALWAYS INCLUDE SECONDS
IT%NAC==:1B9                        ;NO COLON ALLOWED BETWEEN HH AND MM
IT%AAC==:1B10                       ;ALWAYS ALLOW COLON
IT%AMS==:1B11                       ;ALWAYS INTERPRET ONE COLON AS HHMM:SS
IT%AHM==:1B12                       ;ALWAYS INTERPRET ONE COLON AS HH:MM
IT%N24==:1B14                       ;NO 24-HOUR FORMAT
IT%NTM==:1B15                       ;NO TIME MODIFIER (AM, PM)
IT%NTZ==:1B16                       ;NO TIME ZONE


;INLNM

.INLJB==:0                          ;GET JOB WIDE LOGICAL NAME FROM INDEX
.INLSY==:1                          ;GET SYSTEM LOGICAL NAME FROM INDEX
```

```
;IPCF BIT DEFINITIONS AND DATA STRUCTURES

;PACKET FORMAT

.IPCFL==:0                      ;FLAGS WORD
IP%CFB==:1B0                    ;DON'T BLOCK READ
IP%CFS==:1B1                    ;INDIRECT SENDER'S PID
IP%CFR==:1B2                    ;INDIRECT RECEIVER'S PID
IP%CFO==:1B3                    ;OVERDRAW SEND
IP%TTL==:1B4                    ;TRUNCATE ON TOO LARGE MESSAGE
IP%CPD==:1B5                    ;CREATE A PID ON THE SEND
IP%JWP==:1B6                    ;MAKE THE CREATED PID BE JOB WIDE
IP%NOA==:1B7                    ;NO ACCESS OF PID BY OTHER FORKS
IP%CFP==:1B18                   ;SENDER IS PRIV'D AND IS ENVOKING PRIVS
IP%CFV==:1B19                   ;PAGE TRANSFER MODE
IP%CFZ==:1B20                   ;ZERO LENGTH MESSAGE WAS SENT
IP%CFE==:77B29                  ;ERROR FIELD

;ERRORS SENT BY INFO

 .IPCPI==:15                    ;INSUFFICIENT PRIVILEGE
 .IPCUF==:16                    ;ILLEGAL FUNCTION
 .IPCSN==:67                    ;SEND INFO YOUR NAME
 .IPCFF==:72                    ;INFO FREE SPACE EXHAUSTED
 .IPCBP==:74                    ;PID HAS NO NAME OR IS ILLEGAL
 .IPCDN==:75                    ;DUPLICATE NAME
 .IPCNN==:76                    ;UNKNOWN NAME
 .IPCEN==:77                    ;ILLEGAL NAME
IP%CFC==:7B32                   ;SYSTEM SENDER CODE
 .IPCCC==:1                     ;SENT BY [SYSTEM]IPCF
 .IPCCF==:2                     ;SENT BY SYSTEM WIDE [SYSTEM]INFO
 .IPCCP==:3                     ;SENT BY RECEIVER'S [SYSTEM]INFO
IP%CFM==:7B35                   ;SPECIAL MESSAGE RETURN FIELD
 .IPCFN==:1                     ;MESSAGE WAS NOT DELIVERED
.IPCFS==:1                      ;PID OF SENDER
.IPCFR==:2                      ;PID OF RECEIVER
.IPCFP==:3                      ;POINTER TO MESSAGE BLOCK
.IPCFD==:4                      ;LOGGED IN DIR OF SENDER
.IPCFC==:5                      ;ENABLED CAPABILITIES OF SENDER
.IPCSD==:6                      ;CONNECTED DIRECTORY NUMBER OF SENDER
.IPCAS==:7                      ;POINTER TO ACCOUNT STRING OF SENDER
.IPCSU==:26                     ;SPOOL MESSAGE CODE FROM IPCC
.IPCSL==:27                     ;LOGOUT MESSAGE CODE FROM IPCC
.IPCSA==:30                     ;RESOURCE ALLOCATOR MESSAGE CODE
.IPCDS==:31                     ;STRUCTURE DISMOUNT MESSAGE CODE FROM IPCC
.IPCLI==:32                     ;LOGIN MESSAGE CODE FROM IPCC
.IPCLO==:33                     ;LOGOUT MESSAGE TO CREATOR FROM IPCC

.IPCSS==:15                     ;IPCC REQUEST TO INFO TO DELETE PIDS
```

```
;[SYSTEM] INFO DEFINITIONS

.IPCI0==:0                      ;CODE,,FUNCTION
.IPCIW==:1                      ;FIND PID FOR NAME
.IPCIG==:2                      ;FIND NAME FOR PID
.IPCII==:3                      ;ASSIGN NAME TO PID
.IPCIJ==:4                      ;ASSIGN NAME TO PID
.IPCIS==:15                     ;MONITOR DROP PID FUNCTION
.IPCI1==:1                      ;PID TO GET A COPY OF REPLY
.IPCI2==:2                      ;START OF DATA


;JFNS

JS%DEV==7B2                     ;DEVICE  FIELD OUTPUT CONTROL
JS%DIR==:7B5                    ;DIRECTORY FIELD OUTPUT CONTROL
JS%NAM==:7B8                    ;NAME FIELD OUTPUT CONTROL
JS%TYP==:7B11                   ;FILE TYPE FIELD OUTPUT CONTROL
JS%GEN==:7B14                   ;GENERATION FIELD OUTPUT CONTROL
JS%PRO==:7B17                   ;PROTECTION FIELD OUTPUT CONTROL
JS%ACT==:7B20                   ;ACCOUNT FIELD OUTPUT CONTROL
        ;VALUES FOR ABOVE 7 FIELDS:
        .JSNOF==:0              ;NEVER OUTPUT FIELD
        .JSAOF==:1              ;ALWAYS OUTPUT FIELD
        .JSSSD==:2              ;SUPPRESS IF SYSTEM DEFAULT
JS%TMP==:1B21                   ;RETURN ;T IF TEMP FILE
JS%SIZ==:1B22                   ;RETURN SIZE
JS%CDR==:1B23                   ;RETURN CREATION DATE
JS%LWR==:1B24                   ;RETURN LAST WRITE
JS%LRD==:1B25                   ;RETURN LAST READ
JS%PTR==:1B26                   ;AC 2 HOLDS STRING POINTER NOT JFN
JS%ATR==:1B27                   ;RETURN ATTRIBUTES
JS%AT1==:1B28                   ;RETURN 1 SPECIFIC ATTRIBUTE
JS%PSD==:1B32                   ;PUNCTUATE SIZE AND DATE
JS%TBR==:1B33                   ;TAB BEFORE FIELDS RETURNED
JS%TBP==:1B34                   ;TAB BEFORE POSSIBLE FIELDS
JS%PAF==:1B35                   ;PUNCTUATE ALL FIELDS


;LNMST

.LNSJB==:0                      ;GET JOB WIDE DEFINITION OF A LN
.LNSSY==:1                      ;GET SYSTEM DEFINITION OF A LOGICAL NAME


;LOCK

LK%CNT==:1B0                    ;USE COUNT IN AC3
LK%PHY==:1B1                    ;USE AC1 AS PHYSICAL PAGE NUMBER
LK%NCH==:1B2                    ;MAP PAGES CACHE INHIBITED
LK%AOL==:1B3                    ;ALLOW LOCKING IN OFFLINE PAGES
```

```
;MSTR

.MSRNU==:0                        ;READ STATUS OF NEXT DISK UNIT
.MSRUS==:1                        ;READ STATUS OF A DISK UNIT
  .MSRCH==:0                      ;CHANNEL NUMBER
  .MSRCT==:1                      ;CONTROLLER NUMBER
  .MSRUN==:2                      ;UNIT NUMBER
  .MSRST==:3                      ;STATUS
    MS%MNT==:1B0                  ;THIS UNIT IS PART OF A MOUNTED STRUCTURE
    MS%16B==:1B1                  ;THIS UNIT WRITTEN IN 16-BIT MODE
                                  ; (RESERVED FOR FUTURE)
    MS%DIA==:1B2                  ;THIS UNIT IS CURRENTLY IN USE BY AN
                                  ;  ON-LINE DIAGNOSTIC
    MS%OFL==:1B3                  ;THIS UNIT IS OFF-LINE
    MS%ERR==:1B4                  ;THERE WAS AN ERROR READING THIS UNIT
    MS%BBB==:1B5                  ;ONE OF THE BAT BLOCKS IS BAD
    MS%HBB==:1B6                  ;ONE OF THE HOME BLOCKS IS BAD
    MS%WLK==:1B7                  ;UNIT IS WRITE-LOCKED
    MS%TYP==:777B17               ;DISK TYPE CODE
     .MSRP4==:1                   ;RP04
     .MSRP5==:5                   ;RP05
     .MSRP6==:6                   ;RP06
  .MSRSN==:4                      ;STRUCTURE NAME
  .MSRSA==:5                      ;STRUCTURE ALIAS
  .MSRNS==:6                      ;UNIT # IN STRUCTURE,,# OF UNITS IN STRUCTURE
  .MSRSW==:7                      ;NUMBER OF PAGES FOR SWAPPING
  .MSRUI==:10                     ;UNIT ID
  .MSROI==:13                     ;OWNER ID
  .MSRFI==:16                     ;FILE-SYSTEM ID
  .MSRSP==:21                     ;NUMBER OF SECTORS PER PAGE
  .MSRSC==:22                     ;NUMBER OF SECTORS PER CYLINDER
  .MSRPC==:23                     ;NUMBER OF PAGES PER CYLINDER
  .MSRCU==:24                     ;NUMBER OF CYLINDERS PER UNIT
  .MSRSU==:25                     ;NUMBER OF SECTORS PER UNIT
  .MSRBT==:26                     ;NUMBER OF BIT-WORDS IN BIT TABLE PER CYLINDER
  .MSRLN==:27                     ;MAX LENGTH OF ARGUMENT BLOCK IN WORDS

.MSMNT==:2                        ;MOUNT A STRUCTURE
  .MSTNM==:0                      ;NAME OF STRUCTURE
  .MSTAL==:1                      ;ALIAS NAME
  .MSTNU==:2                      ;NUMBER OF UNITS IN STRUCTURE
  .MSTFL==:2                      ;FLAGS (LHS)
    MS%FLG==:777777,,0            ;MASK FOR .MSTFL
    MS%NFH==:1B0                  ;NO FIX BAD HOME BLOCK
    MS%NFB==:1B1                  ;NO FIX BAD BAT BLOCK
    MS%XCL==:1B2                  ;MOUNT FOR EXCLUSIVE USE BY JOB
    MS%IGN==:1B3                  ;IGNORE ERRORS
  .MSTUI==:3                      ;START OF UNIT INFORMATION
     .MSTCH==:0                   ;CHANNEL NUMBER
     .MSTCT==:1                   ;CONTROLLER NUMBER
     .MSTUN==:2                   ;UNIT NUMBER
     .MSTNO==:3                   ;# OF ARGUMENT WORDS/UNIT
```

```
.MSDIS==:3                              ;DISMOUNT A STRUCTURE
  .MSDNM==:0                            ;NAME OF STRUCTURE

.MSGSS==:4                              ;GET STATUS OF A STRUCTURE
  .MSGSN==:0                            ;STRUCTURE NAME (ALIAS)
  .MSGST==:1                            ;STATUS
    MS%PS==:1B0                         ;STRUCTURE IS A PUBLIC STRUCTURE
    MS%DIS==:1B1                        ;STRUCTURE IS BEING DISMOUNTED
    MS%DOM==:1B2                        ;STRUCTURE IS DOMESTIC
    MS%PPS==:1B3                        ;STRUCTURE IS THE PRIMARY PUBLIC STRUCTURE
    MS%INI==:1B4                        ;STRUCTURE IS BEING INITIALIZED
  .MSGNU==:2                            ;NUMBER OF UNITS IN STRUCTURE
  .MSGMC==:3                            ;MOUNT COUNT
  .MSGFC==:4                            ;OPEN FILE COUNT
  .MSGSI==:5                            ;STRUCTURE ID
  .MSGLN==:6                            ;LENGTH OF ARGUMENT BLOCK

.MSSSS==:5                              ;SET STATUS OF A STRUCTURE
  .MSSSN==:0                            ;STRUCTURE NAME
  .MSSST==:1                            ;NEW STATUS BITS
  .MSSMW==:2                            ;MASK WORD OF BITS TO BE CHANGED
  .MSSLN==:3                            ;LENGTH OF ARGUMENT BLOCK

.MSINI==:6                              ;INITIALIZE A STRUCTURE
  .MSINM==:0                            ;NAME OF STRUCTURE
  .MSIAL==:1                            ;ALIAS NAME
  .MSINU==:2                            ;NUMBER OF UNITS IN STRUCTURE
  .MSIFL==:2                            ;FLAGS (LHS)
                                        ;FLAGS DEFINED IN .MSMNT FUNCTION
    MS%FCN==:77B17                      ;FUNCTION CODE
      .MSCRE==:1                        ;CREATE NEW FILE SYSTEM
      .MSRRD==:2                        ;RECONSTRUCT THE ROOT-DIRECTORY
      .MSWHB==:3                        ;WRITE THE HOME BLOCKS
      .MSRIX==:4                                ;REBUILD INDEX TABLE (IDXFIL)
  .MSISU==:3                            ;START OF UNIT INFORMATION
    .MSICH==:0                          ;CHANNEL NUMBER
    .MSICT==:1                          ;CONTROLLER NUMBER
    .MSIUN==:2                          ;UNIT NUMBER
    .MSINO==:3                          ;# OF ARGUMENT WORDS/UNIT
  .MSIST==:6                            ;STATUS WORD
  .MSISW==:7                            ;NUMBER OF PAGES FOR SWAPPING ON THIS UNIT
  .MSIFE==:10                           ;NUMBER OF PAGES FOR FRONT-END FILE SYSTEM
  .MSIUI==:11                           ;UNIT ID
  .MSIOI==:14                           ;OWNER ID
  .MSIFI==:17                           ;FILE SYSTEM ID
  .MSIFB==:22                           ;NUMBER OF PAGES FOR  BOOTSTRAP.BIN (OPTIONAL)

.MSIMC==:7                              ;INCREMENT MOUNT COUNT
.MSDMC==:10                             ;DECREMENT MOUNT COUNT
  .MSDEV==:0                            ;DEVICE DESIGNATOR OR STRUCTURE
```

```
.MSGSU==:11                       ;GET STRUCTURE USERS
  .MSUAL==:0                      ;POINTER TO ALIAS OF STRUCTURE
  .MSUFL==:1                      ;FLAGS,,# OF ITEMS RETURNED
    MS%GTA==:1B0                  ;GET USERS WHO HAVE ACCESSED STRUCTURE
    MS%GTM==:1B1                  ;GET USERS WHO HAVE MOUNTED STRUCTURE
    MS%GTC==:1B2                  ;GET USERS WHO ARE CONNECTED TO STRUCTURE
  .MSUJ1==:2                      ;FIRST JOB NUMBER RETURNED
```

```
;MTOPR - FUNCTION CODES

.MOCLE==:0                          ;CLEAR ERRORS
.MONOP==:31                         ;NOP (WAIT FOR ACTIVITY TO STOP)
.MOREW==:1                          ;REWIND
.MOEOF==:3                          ;WRITE EOF
.MODTE==:4                          ;ASSIGN FE DEVICE TO A DTE
.MOFWR==:6                          ;FORWARD SPACE RECORD
.MOBKR==:7                          ;BACKSPACE RECORD
.MORUL==:11                         ;REWIND AND UNLOAD
.MOERS==:13                         ;ERASE TAPE
.MOFWF==:16                         ;FORWARD SPACE FILE
.MOBKF==:17                         ;BACKSPACE FILE
.MOSPD==:26                         ;SET TTY SPEED (FOR KL ONLY)
.MORSP==:27                         ;READ LINE SPEED (FOR KL ONLY)
  MO%RMT==:1B0                      ;FLAG TO SAY LINE IS REMOTE
  MO%AUT==:1B1                      ;FLAG TO SAY LINE IS "AUTO" SPEED
                                    ; (RSX20F ONLY)
.MOSDR==:2                          ;SET READ DIRECTION
.MORDR==:26                         ;READ READ DIRECTION
.MOEOT==:10                         ;SKIP TO LOGICAL END OF TAPE
.MOSRS==:5                          ;SET RECORD SIZE
.MORRS==:15                         ;READ RECORD SIZE
.MOSDN==:24                         ;SET DENSITY
.MORDN==:12                         ;READ DENSITY
.MOSDM==:4                          ;SET DATA MODE
.MORDM==:14                         ;READ DATA MODE
.MOSPR==:20                         ;SET PARITY
.MORPR==:21                         ;READ PARITY
.MONRB==:22                         ;GET NUMBER OF REMAINING BYTES IN RECORD
.MOFOU==:23                         ;FORCE OUT RECORD
.MOINF==:25                         ;GET INFORMATION ABOUT TAPE
   .MOICT==:0                       ;COUNT OF ARGUMENTS TO BE RETURNED
   .MOITP==:1                       ;MAGTAPE TYPE CODE
      .MTT16==:1                    ;MAGTAPE TYPE TU16
      .MTT45==:2                    ;MAGTAPE TYPE TU45
   .MOIID==:2                       ;MAGTAPE REEL ID
   .MOISN==:3                       ;CHAN,CONTROLLER,UNIT ,, SERIAL #
   .MOIRD==:4                       ;# OF READS DONE
   .MOIWT==:5                       ;# OF WRITES DONE
   .MOIRC==:6                       ;RECORD # FROM BOT
   .MOIFC==:7                       ;FILE COUNT ON TAPE
   .MOISR==:10                      ;# OF SOFT READ ERRORS
   .MOISW==:11                      ;# OF SOFT WRITE ERRORS
   .MOIHR==:12                      ;# OF HARD READ ERRORS
   .MOIHW==:13                      ;# OF HARD WRITE ERRORS
.MOPSI==:27                         ;SET ERROR PSI FOR LPT AND CDR
  MO%MSG==:1B0                      ;SUPPRESS STANDARD CTY MESSAGES
.MOSID==:27                         ;SET REEL I.D.
.MOIEL==:30                         ;INHIBIT ERROR LOGGING
```

```
.MOLVF==:32                       ;LOAD DEVICE'S VFU
.MORVF==:33                       ;READ VFU FILE NAME
.MOLTR==:34                       ;LOAD TRANSLATION RAM
.MORTR==:35                       ;READ RAM FILE NAME
.MOSTS==:36                       ;SET SOFTWARE STATUS
.MORST==:37                       ;READ SOFTWARE STATUS
   MO%LPC==1                      ;PAGE COUNTER OVERFLOW
   MO%LCI==2                      ;CHARACTER INTERRUPT (HARD ERROR)
   MO%LVF==4                      ;VFU ERROR.  PAPER MUST BE RE-ALIGNED
   MO%LVU==20                     ;LINE PRINTER HAS OPTICAL VFU
   MO%RPE==40                     ;RAM PARITY ERROR

   MO%RCK==:1                     ;READ CHECK
   MO%PCK==:2                     ;PICK CHECK
   MO%SCK==:4                     ;STACK CHECK
   MO%HEM==:10                    ;HOPPER EMPTY
   MO%SFL==:20                    ;STACKER FULL

   MO%FNX==:1B17                  ;NON-EXISTENT DEVICE
   MO%OL==:1B16                   ;DEVICE IS OFF-LINE
   MO%HE==:1B15                   ;HARDWARE ERROR
   MO%SER==:1B14                  ;SOFTWARE ERROR
   MO%IOP==:1B13                  ;I/O IN PROGRESS
   MO%EOF==:1B12                  ;END OF FILE
;  1B11                           ;RESERVED
   MO%FER==:1B10                  ;FATAL ERROR
   MO%LCP==:1B0                   ;LOWER CASE PRINTER
   MO%RLD==:1B1                   ;FRONT-END WAS RELOADED
.MOFLO==:40                       ;FLUSH OUTPUT

;SEE SETJB FOR VARIOUS ARGUMENT VALUES

.MOSNT==:34                       ;SET TTY NON-TERMINAL STATUS
   .MOSMN==:1                     ;NO SYSTEM MESSAGES(I.E. SUPPRESS)
   .MOSMY==:0                     ;YES SYSTEM MESSAGES(DEFAULT)
.MORNT==:35                       ;READ TTY NON-TERMINAL STATUS

;PTY MTOPR NUMBERS

.MOAPI==:24                       ;ASSIGN PTY INTERRUPT CHANNELS
   MO%WFI==:1B0                   ;ENABLE WAITING FOR INPUT
   MO%OIR==:1B1                   ;ENABLE OUTPUT IS WAITING
   MO%SIC==:77B17                 ;SOFTWARE INTERRUPT CHANNEL
.MOPIH==:25                       ;TEST PTY INPUT HUNGRY
   .MONWI==:0                     ;NOT WAITING FOR INPUT
   .MOWFI==:-1                    ;WAITING FOR INPUT
.MOBAT==:26                       ;SET BATCH BIT
  .MOJCB==:1                      ;JOB CONTROLLED BY BATCH
  .MONCB==:0                      ;JOB NOT CONTROLLED BY BATCH
```

```
;TTY MODE DEFINITIONS

.MORLW==:30                            ;READ WIDTH
.MOSLW==:31                            ;SET WIDTH
.MORLL==:32                            ;READ LENGTH
.MOSLL==:33                            ;SET LENGTH
.MOSIG==:36                            ;SET "IGNORE INPUT WHEN INACTIVE" BIT

;NET MTOPR NUMBERS

.MOACP==:20           ;TOPS20AN ;ACCEPT CONNECTION ON SOCKET
.MOSND==:21           ;TOPS20AN ;SEND ALL CURENTLY BUFFERED BYTES
.MOSIN==:22           ;TOPS20AN ;SEND INS/INR COMMAND
.MOAIN==:24           ;TOPS20AN ;ASSIGN INS/INR AND FSM PSI CHANNELS
   MO%NIN==:77B5      ;TOPS20AN ;INS/INR SOFTWARE INTERRUPT CHANNEL
   MO%FSM==:77B17     ;TOPS20AN ;FSM CHANGE OF STATE INTERRUPT CHANNEL


;DEFINITIONS FOR DECNET

.MOACN==:24                            ;ASSIGN CONNECT INTERRUPT CHANNEL
   MO%CDN==:777B8                      ;CONNECT INTERRUPT CHANNEL
   MO%INA==:777B17                     ;INTERRUPT MESSAGE CHANNEL
   MO%DAV==:777B26                     ;DATA AVAILABLE CHANNEL
     .MONCI==:777                      ;NO CHANGE
     .MOCIA==:776                      ;CLEAR INTERRUPT ASSIGNMENT

.MORLS==:25                            ;READ LINK STATUS
   MO%CON==:1B0                        ;LINK IS CONNECTED
   MO%SRV==:1B1                        ;LINK IS A SERVER
   MO%WFC==:1B2                        ;WAITING FOR A CONNECT
   MO%WCC==:1B3                        ;WAITING FOR THIS LINK TO CONFIRM
   MO%EOM==:1B4                        ;EOM PRESENT IN INPUT BUFFER
   MO%ABT==:1B5                        ;CONNECTION ABORTED
   MO%SYN==:1B6                        ;SYNCH DI RECIEVED
   MO%INT==:1B7                        ;INT MESSAGE AVAILABLE
.MORHN==:26                            ;READ HOST NAME
.MORTN==:27                            ;READ TASK NAME
.MORUS==:30                            ;READ USER DATA
.MORPW==:31                            ;READ PASSWORD
.MORAC==:32                            ;READ ACCOUNT
.MORDA==:33                            ;READ OPTIONAL DATA
.MORCN==:34                            ;READ CONNECT OBJECT NUMBER
.MORIM==:35                            ;READ INTERRUPT MESSAGE
.MOSIM==:36                            ;SEND INTERRUPT MESSAGE
.MOROD==:37                            ;READ OBJ-DESC OF CONNECTION
.MOCLZ==:40                            ;CLOSE/REJECT A CONNECTION
.MOCC==:41                             ;ACCEPT A CONNECTION
.MORSS==:42                            ;READ SEGMENT SIZE
```

```
;MUTIL JSYS FUNCTION CODES

.MUENB==:1                      ;ENABLE PID FOR RECEIVING
.MUDIS==:2                      ;DISABLE PID FROM RECEIVING
.MUGTI==:3                      ;GET PID OF [SYSTEM]INFO
.MUCPI==:4                      ;CREATE A PRIVATE INFO FOR A JOB
.MUDES==:5                      ;DESTROY A PID
.MUCRE==:6                      ;CREATE A PID
.MUSSQ==:7                      ;SET SEND AND RECEIVE QUOTAS
.MUCHO==:10                     ;CHANGE OWNER OF A PID
.MUFOJ==:11                     ;FIND OWNER'S JOB NUMBER
.MUFJP==:12                     ;FIND JOB'S PIDS
.MUFSQ==:13                     ;FIND SEND AND RECEIVE QUOTAS
.MUFFP==:15                     ;FIND FORK'S PIDS
.MUSPQ==:16                     ;SET PID QUOTA
.MUFPQ==:17                     ;FIND PID QUOTA
.MUQRY==:20                     ;QUERY
.MUAPF==:21                     ;ASSOCIATE A PID WITH A FORK
.MUPIC==:22                     ;PUT PID ON AN INTERRUPT CHANNEL
.MUDFI==:23                     ;DEFINE PID OF [SYSTEM]INFO
.MUSSP==:24                     ;SET SYSTEM PID TABLE
.MURSP==:25                     ;READ SYSTEM PID TABLE
.MUMPS==:26                     ;GET MAXIMUM PACKET SIZE


;SYSTEM PID TABLE INDEX VALUES

.SPIPC==:0                      ;PID OF IPCC
.SPINF==:1                      ;PID OF INFO
.SPQSR==:2                      ;PID OF QUASAR
.SPMDA==:3                      ;PID OF QSRMDA
.SPOPR==:4                      ;PID OF OPERATOR JOB (ORION)


;NODE

.NDSLN==0                       ;SET LOCAL NODE NAME
.NDGLN==1                       ;GET LOCAL NODE NAME
   .NDNOD==0                    ;POINTER TO NODE NAME

;NOUT

NO%MAG==:1B0                    ;OUTPUT MAGNITUDE
NO%SGN==:1B1                    ;OUTPUT SIGN
NO%LFL==:1B2                    ;LEADING FILLER
NO%ZRO==:1B3                    ;FILL WITH ZERO'S
NO%OOV==:1B4                    ;OUTPUT ON COLUMN OVERFLOW
NO%AST==:1B5                    ;OUTPUT ASTERISKS ON OVERFLOW
NO%COL==:177B17                 ;NUMBER OF COLUMNS TO USE
NO%RDX==:777777                 ;RADIX
```

```
;ODCNV -- SEE IDCNV FOR BITS


;ODTIM

OT%NDA==:1B0                     ;DO NOT OUTPUT DATE
OT%DAY==:1B1                     ;OUTPUT DAY OF WEEK
OT%FDY==:1B2                     ;OUTPUT NUMERIC MONTH
OT%NMN==:1B3                     ;OUTPUT NUMERIC MONTH
OT%FMN==:1B4                     ;OUTPUT MONTH IN FULL
OT%4YR==:1B5                     ;OUTPUT 4-DIGIT YEAR
OT%DAM==:1B6                     ;OUTPUT DAY AFTER MONTH
OT%SPA==:1B7                     ;OUTPUT SPACES IN DATE
OT%SLA==:1B8                     ;OUTPUT SLASHES IN DATE
OT%NTM==:1B9                     ;DO NOT OUTPUT TIME
OT%NSC==:1B10                    ;DO NOT OUTPUT SECONDS
OT%12H==:1B11                    ;OUTPUT 12-HOUR FORMAT
OT%NCO==:1B12                    ;DO NOT OUTPUT COLON
OT%TMZ==:1B13                    ;OUTPUT TIME ZONE
OT%SCL==:1B17                    ;SUPPRESS COLUMNIZATION


;ODTNC -- SEE IDCNV FOR BITS


;OPENF

OF%BSZ==:77B5                    ;BYTE SIZE
OF%MOD==:17B9                    ;MODE
OF%HER==:1B18                    ;HALT ON IO ERROR
OF%RD==:1B19                     ;READ
OF%WR==:1B20                     ;WRITE
OF%EX==:1B21                     ;EXECUTE (RESERVED FOR THE FUTURE)
OF%APP==:1B22                    ;APPEND
OF%THW==:1B25                    ;THAWED
OF%AWT==:1B26                    ;ALWAYS WAIT
OF%PDT==:1B27                    ;PRESERVE DATES
OF%NWT==:1B28                    ;NEVER WAIT
OF%RTD==:1B29                    ;RESTRICTED
OF%PLN==:1B30                    ;SET TO DISABLE LINE NUMBER CHECKING FOR
                                 ; NON-LINE NUMBER FILES
OF%DUD==:1B31                    ;DON'T UPDATE TO DISK BY DDMP
OF%OFL==:1B32                    ;ALLOW OPENING THE DEVICE EVEN IF OFFLINE
```

```
;PMAP BIT DEFINITIONS

PM%CNT==:1B0                      ;RH WORD CONTAINS A COUNT
PM%MVP==:1B1                      ;MOVE PAGE INSTEAD OF INDIRECT POINTER
                                  ; (NOT IMPLEMENTED -- OBSOLETE)
PM%RD==:1B2                       ;READ
PM%WT==:1B3                       ;WRITE
PM%WR==:1B3                       ; (ANOTHER NAME FOR ABOVE)
PM%EX==:1B4                       ;EXECUTE (RESERVED FOR THE FUTURE)
PM%RWX==:7B4                      ;CONVENIENT ABBREV FOR RD+WT+EX
PM%PLD==:1B5                      ;PRELOAD PAGES BEING MAPPED
PM%IND==:1B6                      ;USE INDIRECT PTRS (RESERVED FOR THE FUTURE)
PM%TPU==:1B8                      ;TRAP TO USER
                                  ; (NOT IMPLEMENTED -- OBSOLETE)
PM%CPY==:1B9                      ;COPY ON WRITE
PM%RPT==:777777B35                ;REPEAT COUNT


;PMCTL - PHYSICAL MEMORY CONTROL

.MCRCE==:0                        ;READ CACHE ENABLE
.MCSCE==:1                        ;SET CACHE ENABLE
 .MCCST==:0                       ;ARGLIST OFFSET FOR CACHE STATE
 MC%CEN==:1                       ;CACHE ENABLED
.MCRPS==:2                        ;READ PAGE STATUS
.MCSPS==:3                        ;SET PAGE STATUS
 .MCPPN==:0                       ;ARGLIST OFFSET FOR PHYSICAL PAGE NUMBER
 .MCPST==:1                       ;ARGLIST OFFSET FOR PAGE STATE
 .MCPSA==:0                       ;PAGE AVAILABLE
 .MCPSS==:1                       ;PAGE IN TRANSITION STATE
 .MCPSO==:2                       ;PAGE OFFLINE
 .MCPSE==:3                       ;PAGE OFFLINE DUE TO ERROR
.MCRME==:4                        ;READ MEMORY ERROR INFORMATION


;PRARG - PROCESS ARGUMENTS

;FUNCTION CODE DEFINITIONS

.PRARD==:1                        ;READ ARGUMENT BLOCK
.PRAST==:2                        ;SET ARGUMENT BLOCK
```

```
;RCUSR AND RCDIR

; FLAGS SUPPLIED ON CALL

RC%PAR==:1B14                   ;PARTIAL RECOGNITION IS ALLOWED
RC%STP==:1B15                   ;STEP WILDCARD (RCDIR ONLY)
RC%AWL==:1B16                   ;ALLOW WILDCARDS (RCDIR ONLY)
RC%EMO==:1B17                   ;EXACT MATCH ONLY

; FLAGS RETURNED

RC%DIR==1B0                     ;FILES-ONLY DIRECTORY
RC%ANA==1B1                     ;ALPHANUMERIC ACCOUNTS ALLOWED
RC%RLM==1B2                     ;REPEAT LOGIN MESSAGE
RC%NOM==:1B3                    ;NO MATCH FOUND
RC%AMB==:1B4                    ;AMBIGUOUS
RC%NMD==:1B5                    ;NO MORE DIRS - RETURNED IF STP IS REQUESTED
RC%WLD==:1B6                    ;WILDCARD DIR WAS INPUT


;RDTTY AND TEXTI

RD%BRK==:1B0                    ;BREAK ON REGULAR BREAK SET
RD%TOP==:1B1                    ;BREAK ON TOPS10 BREAK SET
RD%PUN==:1B2                    ;BREAK ON PUNCTUATION
RD%BEL==:1B3                    ;BREAK ON END OF LINE
RD%CRF==:1B4                    ;SUPPRESS CR (RETURNS LF ONLY)
RD%RND==:1B5                    ;RETURN IF NOTHING TO DELETE
RD%JFN==:1B6                    ;JFNS GIVEN FOR SOURCE
RD%RIE==:1B7                    ;RETURN ON INPUT (BUFFER) EMPTY
RD%BBG==:1B8                    ;BEGINNING OF (DEST) BUFFER GIVEN
RD%RBF==:1B9                    ;^R BUFFER IS DISJOINT
RD%RAI==:1B10                   ;RAISE LOWERCASE INPUT
RD%SUI==:1B11                   ;SUPPRESS ^U INDICATION
RD%BTM==:1B12                   ;BREAK CHARACTER TERMINATED INPUT
RD%BFE==:1B13                   ;RETURNED BECAUSE BUFFER EMPTY
RD%BLR==:1B14                   ;BACKUP LIMIT REACHED

;TEXTI ARG BLOCK

.RDCWB==:0                      ;COUNT OF WORDS IN BLOCK
.RDFLG==:1                      ;FLAGS
.RDIOJ==:2                      ;IO JFNS
.RDDBP==:3                      ;DEST BYTE POINTER
.RDDBC==:4                      ;DEST BYTE COUNT
.RDBFP==:5                      ;TOP OF BUFFER POINTER
.RDRTY==:6                      ;RETYPE (^R) POINTER
.RDBRK==:7                      ;BREAK SET MASK POINTER
.RDBKL==:10                     ;BACKUP LIMIT POINTER
```

```
;RFSTS

RF%FRZ==:1B0                    ;PROCESS IS FROZEN
RF%STS==:377777B17              ;PROCESS STATUS CODE
   .RFRUN==:0                   ;RUNNABLE
   .RFIO==:1                    ;DISMISSED FOR I/O
   .RFHLT==:2                   ;HALTED
   .RFFPT==:3                   ;FORCED PROCESS TERMINATION
   .RFWAT==:4                   ;WAITING FOR INFERIOR PROCESS
   .RFSLP==:5                   ;SLEEP
   .RFTRP==:6                   ;JSYS TRAPPED
   .RFABK==:7                   ;ADDRESS BREAK FREEZE
RF%SIC==:777777B35              ;SOFTWARE INTERRUPT CHANNNEL


;RFTAD/SFTAD

.RSWRT==:0                      ;WRITE DATE WORD
.RSCRV==:1                      ;CREATION DATE WORD
.RSREF==:2                      ;REFERENCE DATE WORD
.RSCRE==:3                      ;INTERNAL SYSTEM WRITE DATE WORD


;RMAP

RM%RD==1B2                      ;READ ACCESS ALLOWED
RM%WR==:1B3                     ;WRITE ACCESS ALLOWED
RM%EX==:1B4                     ;EXECUTE ACCESS ALLOWED
RM%PEX==:1B5                    ;PAGE EXISTS
RM%CPY==:1B9                    ;COPY ON WRITE


;RPACS/SPACS BIT DEFINITIONS

PA%RD==:1B2                     ;READ ACCESS ALLOWED
PA%WT==:1B3                     ;WRITE ACCESS ALLOWED
PA%WR==:1B3                     ; (ANOTHER NAME FOR ABOVE)
PA%EX==:1B4                     ;EXECUTE ACCESS ALLOWED
                                ; (RESERVED FOR THE FUTURE)
PA%PEX==:1B5                    ;PAGE EXISTS
PA%IND==:1B6                    ;INDIRECT POINTER
PA%TPU==:1B8                    ;TRAP TO USER
                                ; (NOT IMPLEMENTED -- OBSOLETE)
PA%CPY==:1B9                    ;COPY ON WRITE
PA%PRV==:1B10                   ;PRIVATE
P1%RD==:1B20                    ;READ ACCESS ALLOWED IN 1ST POINTER
P1%WR==:1B21                    ;WRITE ACCESS ALLOWED IN 1ST POINTER
P1%WT==:1B21                    ; (ANOTHER NAME FOR ABOVE)
P1%EX==:1B22                    ;EXECUTE ACCESS ALLOWED IN 1ST POINTER
                                ; (RESERVED FOR THE FUTURE)
P1%PEX==:1B23                   ;PAGE EXISTS IN 1ST POINTER
P1%CPY==:1B27                   ;COPY-ON-WRITE IN 1ST POINTER
```

```
;RSCAN

.RSINI==:0                          ;MAKE RESCAN BUFFER AVAILABLE FOR INPUT
.RSCNT==:1                          ;COUNT CHARACTERS LEFT TO READ FROM RESCAN BUFFER


;RTIW

RT%DIM==:1B0                        ;DEFERRED TERMINAL INTERRUPT MASK GIVEN
RT%PRH==:377777B35                  ;PROCESS HANDLE


;SCTTY

.SCRET==:0                          ;RETURN DESIGNATOR (CTTY) FOR FORK
.SCSET==:1                          ;SET SCTTY FOR FORK
.SCRST==:2                          ;CLEAR FORK CTTY (RESTORE JOB CTTY)

;SCVEC

.SVEAD==:0                          ;ENTRY ADDRESS
.SVINE==:1                          ;INITIAL ENTRY FOR SETUP
.SVGET==:2                          ;ENTRY ADDRESS FOR GET SHARE FILE ROUTINE
.SV40==:3                           ;ADDRESS TO GET LOCATION 40
.SVRPC==:4                          ;ADDRESS TO GET RETURN PC
.SVMAK==:5                          ;ENTRY FOR MAKE SHARE FILE ROUTINE
.SVCST==:6                          ;2 WORD BLOCK FOR CONTROL-C/START PROCESSING


;SDVEC

.SDEAD==:0                          ;ENTRY ADDRESS
.SDINE==:1                          ;INITIAL ENTRY
.SDVER==:2                          ;DMS VERSION
.SDDMS==:3                          ;ADDRESS TO STORE DMS JSYS
.SDRPC==:4                          ;ADDRESS TO STORE RETURN PC
```

```
;SETJB FUNCTION CODES

.SJDEN==:0                       ;SET DEFAULT MAGTAPE DENSITY
 .SJDDN==:0                      ;SYSTEM DEFAULT DENSITY
 .SJDN2==:1                      ;200 BPI
 .SJDN5==:2                      ;556 BPI
 .SJDN8==:3                      ;800 BPI
 .SJD16==:4                      ;1600 BPI
.SJPAR==:1                       ;SET DEFAULT MAGTAPE PARITY
 .SJPRO==:0                      ;ODD PARITY
 .SJPRE==:1                      ;EVEN PARITY
.SJDM==:2                        ;SET DEFAULT MAGTAPE DATA MODE
 .SJDDM==:0                      ;SYSTEM DEFAULT DATA MODE
 .SJDMC==:1                      ;CORE DUMP MODE
 .SJDM6==:2                      ;SIX BIT BYTE MODE (FOR 7-TRACK DRIVES)
 .SJDMA==:3                      ;ANSI ASCII MODE (7 BITS IN 8 BIT BYTE)
 .SJDM8==:4                      ;INDUSTRY COMPATIBLE MODE
.SJRS==:3                        ;SET DEFAULT MAGTAPE RECORD SIZE
.SJDFS==:4                       ;SET DEFERRED SPOOLING
 .SJSPI==:0                      ;IMMEDIATE MODE SPOOLING
 .SJSPD==:1                      ;DEFERRED MODE SPOOLING
.SJSRM==:5                       ;SET JOB SESSION REMARK


;SFUST

.SFAUT==:0                       ;SET AUTHOR STRING
.SFLWR==:1                       ;SET LAST WRITER STRING
```

```
;SMON FUNCTION CODES AND BIT DEFINITIONS (SYSTEM FLAGS)

.SFFAC==:0                          ;ALLOW FACT ENTRIES
.SFCDE==:1                          ;CHECKDISK FOUND ERRORS
.SFCDR==:2                          ;CHECKDISK RUNNING
.SFMST==:3                          ;MANUAL START IN PROGRESS
.SFRMT==:4                          ;REMOTE LOGINS ALLOWED
.SFPTY==:5                          ;PTY LOGINS ALLOWED
.SFCTY==:6                          ;CTY LOGIN ALLOWED
.SFOPR==:7                          ;OPERATOR IN ATTENDANCE
.SFLCL==:10                         ;LOCAL LOGINS ALLOWED
.SFBTE==:11                         ;BIT TABLE ERRORS FOUND ON STARTUP
.SFCRD==:12                         ;USER CAN CHANGE DIRECTORY CHARACTERISTICS
.SFNVT==:13               ;TOPS20AN ;NVT LOGIN ALLOWED
.SFWCT==:14                         ;WHEEL LOGIN ON CTY ALLOWED
.SFWLC==:15                         ;WHEEL LOGIN ON LOCAL TERMINALS ALLOWED
.SFWRM==:16                         ;WHEEL LOGIN ON REMOTE TERMINALS ALLOWED
.SFWPT==:17                         ;WHEEL LOGIN ON PTY'S ALLOWED
.SFWNV==:20               ;TOPS20AN ;WHEEL LOGIN ON NVT'S ALLOWED
.SFUSG==:21                         ;USAGE FILE IN USE
.SFFLO==:22                         ;FULL LATENCY OPTIMIZATION
                                    ;CAUTION: SETTING THIS REQUIRES THAT THE
                                    ;  SYSTEM BE AT REVISION LEVEL 10, AND
                                    ;  THAT RH20 BOARD M8555 BE AT REVISION LEVEL D.
                                    ;  OTHERWISE, THE FILE-SYSTEM MAY BE DAMAGED.

;BELOW ARE FUNCTION CODES WHICH DO NOT MAP DIRECTLY INTO BITS

.SFNTN==:44               ;TOPS20AN ;NETWORK ON/OFF CONTROL
.SFNDU==:45               ;TOPS20AN ;NET DOWN/UP REQUEST
.SFNHI==:46               ;TOPS20AN ;NET HOST TABLE INITIALIZE
.SFTMZ==:47                         ;SET TIME ZONE THIS SYSTEM IS IN
.SFLHN==:50               ;TOPS20AN ;SET LOCAL HOST NUMBER OF THIS NET SITE
.SFAVR==:51                         ;ACCOUNT VALIDATION ON/OFF
SF%FAC==:1B<.SFFAC>                 ;FACT ENTRIES ALLOWED
SF%CDE==:1B<.SFCDE>                 ;CHECKDISK FOUND ERRORS
SF%CDR==:1B<.SFCDR>                 ;CHECKDISK RUNNING
SF%MST==:1B<.SFMST>                 ;MANUAL START IN PROGRESS
SF%RMT==:1B<.SFRMT>                 ;REMOTE LOGINS ALLOWED
SF%PTY==:1B<.SFPTY>                 ;PTY LOGINS ALLOWED
SF%CTY==:1B<.SFCTY>                 ;CTY LOGIN ALLOWED
SF%OPR==:1B<.SFOPR>                 ;OPERATOR IN ATTENDANCE
SF%LCL==:1B<.SFLCL>                 ;LOCAL LOGINS ALLOWED
SF%BTE==:1B<.SFBTE>                 ;BIT TABLE ERRORS FOUND ON STARTUP
SF%CRD==:1B<.SFCRD>                 ;USER CAN CHANGE DIRECTORY CHARACTERISTICS
SF%NVT==:1B<.SFNVT>       ;TOPS20AN ;NVT LOGINS ALLOWED
SF%USG==:1B<.SFUSG>                 ;USAGE FILE IN USE
SF%FLO==:1B<.SFFLO>                 ;FULL LATENCY OPTIMIZATION IN USE
                                    ;CAUTION: SETTING THIS REQUIRES THAT THE
                                    ;  SYSTEM BE AT REVISION LEVEL 10, AND
                                    ;  THAT RH20 BOARD M8555 BE AT REVISION LEVEL D.
                                    ;  OTHERWISE, THE FILE-SYSTEM MAY BE DAMAGED.
```

```
;SNOOP JSYS DEFINITIONS

;SNOOP FUNCTION CODES

.SNPLC==:0                        ;LOCK CODE INTO MONITOR VIRT MEMORY
.SNPLS==:1                        ;LOCK DOWN THE SWAPPABLE MONITOR
.SNPDB==:2                        ;DEFINE A BREAK POINT
.SNPIB==:3                        ;INSERT THE BREAK POINTS
.SNPRB==:4                        ;REMOVE THE BREAK POINTS
.SNPUL==:5                        ;UNLOCK AND RELEASE ALL SNOOP RESOURCES
.SNPSY==:6                        ;LOOK UP A MONITOR SYMBOL
.SNPAD==:7                        ;LOOK UP ADDRESS IN SYMBOL TABLE


;SPOOL JSYS FUNCTION CODES

.SPLDI==:0                        ;DEFINE AN INPUT SPOOLING DEVICE
.SPLSD==:1                        ;SET DIRECTORY OF SPOOLED DEVICE
.SPLRD==:2                        ;READ DIRECTORY OF SPOOLED DEVICE

;FLAGS IN SPOOL MESSAGE ON LOGOUT AND SPOOLED FILE CLOSE

SP%BAT==:1B0                      ;JOB IS A BATCH JOB
SP%DFS==:1B1                      ;SPOOLING IS DEFERRED
SP%ELO==:1B2                      ;JOB EXECUTED LGOUT JSYS ITSELF
SP%FLO==:1B3                      ;JOB FORCED TO LOG OUT BY TRAP IN TOP FK
SP%OLO==:1B4                      ;OTHER JOB AIMED LGOUT AT THIS ONE

;SPOOL ARGUMENT BLOCK

.SPLDV==:0                        ;DEVICE DESIGNATOR
.SPLNA==:1                        ;NAME STRING
.SPLDR==:1                        ;DIRECTORY NUMBER
.SPLGN==:2                        ;GENERATION NUMBER


;SSAVE

SS%NNP==777777B17                 ;NEGATIVE NUMBER OF PAGES
SS%CPY==:1B18                     ;ALLOW COPY-ON-WRITE
SS%UCA==:1B19                     ;USE CURRENT ACCESS
SS%RD==:1B20                      ;ALLOW READ ACCESS
SS%WR==:1B21                      ;ALLOW WRITE ACCESS
SS%EXE==:1B22                     ;ALLOW EXECUTE ACCESS
SS%FPN==:777B37                   ;FIRST PAGE NUMBER


;STCMP

SC%LSS==:1B0                      ;T1 LESS THAN T2
SC%SUB==:1B1                      ;T1 SUBSTRING OF T2
SC%GTR==:1B2                      ;T1 GREATER THAN T2
```

```
;STDIR

ST%DIR==:1B0                    ;FILES ONLY DIRECTORY
ST%ANA==:1B1                    ;ALPHANUMERIC ACCOUNTS
ST%RLM==:1B2                    ;REPEAT LOGIN MESSAGE


;STIW

ST%DIM==:1B0                    ;SET DEFERRED INTERRUPT MASK
ST%PRH==:777777B35              ;PROCESS HANDLE


;TBLUK

TL%NOM==:1B0                    ;NO MATCH
TL%AMB==:1B1                    ;AMBIGUOUS
TL%ABR==:1B2                    ;LEGAL ABBREVIATION
TL%EXM==:1B3                    ;EXACT MATCH


;TFORK

;FUNCTION CODES IN LH AC1

.TFSET==:0                      ;SET TRAPS AS SPEC'D BY BIT TABLE
.TFRAL==:1                      ;REMOVE ALL TRAPS SET BY THIS FORK
.TFRTP==:2                      ;REMOVE TRAPS SET BY THIS FORK
.TFSPS==:3                      ;SET JSYS TRAP PSI CHAN IN LH(2)
.TFRPS==:4                      ;READ JSYS TRAP PSI CHAN INTO LH(2)
.TFTST==:5                      ;TEST IF SELF MONITORED
.TFRES==:6                      ;REMOVE TRAPS FROM ALL INFERIORS, CLR PSI
.TFUUO==:7                      ;SET UUO TRAPS FOR FORK
.TFSJU==:8                      ;SET BOTH UUO AND JSYS TRAPS
.TFRUU==:9                      ;REMOVE UUO TRAPS

;TIMER DEFINITIONS

.TIMRT==:0                      ;SET TIME LIMIT
.TIMEL==:1                      ;SET ELAPSED TIME CLOCK
.TIMDT==:2                      ;SET DATE & TIME CLOCK
.TIMDD==:3                      ;DELETE AN EXPLICT DATE & TIME CLOCK
.TIMBF==:4                      ;DELETE ALL ENTIRES BEFORE D&T
.TIMAL==:5                      ;DELETE ALL (INCLUDES TIME LIMIT)
```

```
;TLINK

TL%CRO==:1B0                    ;CLEAR REMOTE TO OBJECT LINK
TL%COR==:1B1                    ;CLEAR OBJECT TO REMOTE LINK
TL%EOR==:1B2                    ;ESTABLIST OBJECT TO REMOTE LINK
TL%ERO==:1B3                    ;ESTABLISH REMOTE TO OBJECT LINK
TL%SAB==:1B4                    ;SET ACCEPT BIT FOR OBJECT
TL%ABS==:1B5                    ;ACCEPT BIT STATE
TL%STA==:1B6                    ;SET OR CLEAR ADVICE
TL%AAD==1B7                     ;ACCEPT ADVICE
TL%OBJ==:777777B35              ;OBJECT DESIGNATOR


;UTEST FUNCTION CODES

.UTSET==:0                      ;START TESTING
.UTCLR==:1                      ;STOP TESTING AND RETURN RESULTS

;UTEST ARGUMENT BLOCK

.UTADR==:0                      ;STARTING ADDRESS OF CODE
.UTLEN==:1                      ;LENGTH OF CODE
.UTMAP==:2                      ;START OF BIT MAP

;USAGE

.USENT==:0                      ;WRITE ENTRY
.USCLS==:1                      ;CLOSE OUT CURRENT FILE
.USCKP==:2                      ;PERFORM CHECKPOINT
.USLGI==:3                      ;LOGIN
.USLGO==:4                      ;LOGOUT
.USSEN==:5                      ;SESSION END
.USCKI==:6                      ;SET CHECKPOINT INTERVAL
.USENA==:7                      ;ENABLE ACCOUNT VALIDATION

;UTFRK

UT%TRP==:1B0                    ;ITRAP (OR DO ERJMP/ERCAL) TRAPPED JSYS
```

```
;************************************************
;GENERAL FIELD AND VALUE DEFINITIONS
;USED BY MANY JSYSES
;************************************************


;GENERAL FORK HANDLES

.FHSLF==:400000                 ;SELF
.FHSUP==:<Z -1>                 ;SUPERIOR
.FHTOP==:<Z -2>                 ;TOP IN JOB
.FHSAI==:<Z -3>                 ;SELF AND INFERIORS
.FHINF==:<Z -4>                 ;INFERIORS
.FHJOB==:<Z -5>                 ;ALL IN JOB


;FIELDS OF JFN MODE WORD

TT%OSP==:1B0                    ;OUTPUT SUPPRESS
TT%MFF==:1B1                    ;MECHANICAL FORMFEED PRESENT
TT%TAB==:1B2                    ;MECHANICAL TAB PRESENT
TT%LCA==:1B3                    ;LOWER CASE CAPABILITIES PRESENT
TT%LEN==:177B10                 ;PAGE LENGTH
TT%WID==:177B17                 ;PAGE WIDTH
TT%WAK==:77B23                  ;WAKEUP FIELD
TT%WK0==:1B18                   ;WAKEUP CLASS 0 (UNUSED)
TT%WK1==:1B19                   ;WAKEUP CLASS 1 (UNUSED)
TT%WKF==:1B20                   ;WAKEUP ON FORMATTING CONTROL CHARS
TT%WKN==:1B21                   ;WAKEUP ON NON-FORMATTING CONTROLS
TT%WKP==:1B22                   ;WAKEUP ON PUNCTUATION
TT%WKA==:1B23                   ;WAKEUP ON ALPHANUMERICS
TT%ECO==:1B24                   ;ECHOS ON
TT%ECM==:1B25                   ;ECHO MODE
TT%ALK==:1B26                   ;ALLOW LINKS
TT%AAD==:1B27                   ;ALLOW ADVICE (NOT IMPLEMENTED)
TT%DAM==:3B29                   ;DATA MODE
.TTBIN==:0                      ;BINARY
.TTASC==:1                      ;ASCII
.TTATO==:2                      ;ASCII AND TRANSLATE OUTPUT ONLY
.TTATE==:3                      ;ASCII AND TRANSLATE ECHOES ONLY
TT%UOC==:1B30                   ;UPPER CASE OUTPUT CONTROL
TT%LIC==:1B31                   ;LOWER CASE INPUT CONTROL
TT%DUM==:3B33                   ;DUPLEX MODE
.TTFDX==:0                      ;FULL DUPLEX
.TT0DX==:1                      ;NOT USED, RESERVED
.TTHDX==:2                      ;HALF DUPLEX (CHARACTER)
.TTLDX==:3                      ;LINE HALF DUPLEX
TT%PGM==:1B34                   ;PAGE MODE
TT%CAR==:1B35                   ;CARRIER STATE
```

```
;DIRECTORY PROTECTION DEFINITIONS (3 6-BIT FIELDS: OWNER, GROUP, WORLD)

DP%RD==:40                      ;READING DIRECTORY IS ALLOWED
DP%CN==:10                      ;CONNECT TO DIR, OR CHANGE PROT/ACCOUNT
DP%CF==:4                       ;CREATING FILES IN DIR IS ALLOWED

;FILE PROTECTION DEFINITIONS (3 6-BIT FIELDS: OWNER, GROUP, WORLD)

FP%DIR==:2                      ;DIRECTORY LISTING
FP%APP==:4                      ;APPEND
FP%EX==:10                      ;EXECUTE
FP%WR==:20                      ;WRITE
FP%RD==:40                      ;READ


;INPUT AND OUTPUT IDENTIFIERS

.PRIIN==:100                    ;PRIMARY INPUT
.PRIOU==:101                    ;PRIMARY OUTPUT
.NULIO==:377777                 ;NULL DESIGNATOR
.CTTRM==:777777                 ;JOB'S CONTROLLING TERMINAL
.DVDES==:600000                 ;UNIVERSAL DEVICE CODE
.TTDES==:400000                 ;UNIVERSAL TERMINAL CODE


;MAGTAPE DEVICE STATUS BITS

MT%ILW==:1B18                   ;ILLEGAL WRITE
MT%DVE==:1B19                   ;DEVICE ERROR
MT%DAE==:1B20                   ;DATA ERROR
MT%SER==:1B21                   ;SUPPRESS ERROR RECOVERY PROCEDURES
MT%EOF==:1B22                   ;EOF (FILE MARK)
MT%IRL==:1B23                   ;INCORRECT RECORD LENGTH
MT%BOT==:1B24                   ;BEGINNING OF TAPE
MT%EOT==:1B25                   ;END OF TAPE
MT%EVP==:1B26                   ;EVEN PARITY
MT%DEN==:3B28                   ;DENSITY (0 IS 'NORMAL')
.MTLOD==:1                      ;LOW DENSITY (200 BPI)
.MTMED==:2                      ;MEDIUM DENSITY (556 BPI)
.MTHID==:3                      ;HIGH DENSITY (800 BPI)
MT%CCT==:7B31                   ;CHARACTER COUNTER


;DEVICE DATA MODES

.DMASC==:1                      ;ASCII
.DMIMG==:10                     ;IMAGE
.DMIMB==:13                     ;IMAGE BINARY
.DMBIN==:14                     ;BINARY
```

```
;DEFINED PSI CHANNELS

RADIX 5+5

.ICAOV==:6                          ;ARITHMETIC OVERFLOW
.ICFOV==:7                          ;FLOATING OVERFLOW
.ICPOV==:9                          ;PDL OVERFLOW
.ICEOF==:10                         ;END OF FILE
.ICDAE==:11                         ;DATA ERROR
.ICQTA==:12                         ;QUOTA/DISK EXCEEDED
.ICTOD==:14                         ;TIME OF DAY (NOT IMPLEMENTED)
.ICILI==:15                         ;ILLEG INSTRUCTION
.ICIRD==:16                         ;ILLEGAL READ
.ICIWR==:17                         ;ILLEGAL WRITE
.ICIEX==:18                         ;ILLEGAL EXECUTE (NOT IMPLEMENTED)
.ICIFT==:19                         ;INFERIOR FORK TERMINATION
.ICMSE==:20                         ;MACHINE SIZE EXCEEDED
.ICTRU==:21                         ;TRAP TO USER (NOT IMPLEMENTED)
.ICNXP==:22                         ;NONEXISTENT PAGE REFERENCED
```

```
;TERMINAL TYPE NUMBERS

.TT33==:0                       ;MODEL 33
.TT35==:1                       ;MODEL 35
.TT37==:2                       ;MODEL 37
.TTEXE==:3                      ;EXECUPORT
.TTDEF==:^D8                    ;DEFAULT
.TTIDL==:^D9                    ;IDEAL
.TTV05==:^D10                   ;VT05
.TTV50==:^D11                   ;VT50
.TTL30==:^D12                   ;LA30
.TTG40==:^D13                   ;GT40
.TTL36==:^D14                   ;LA36
.TTV52==:^D15                   ;VT52

;DEFINED TERMINAL CODES

.TICBK==:0                      ;BREAK
.TICCA==:1                      ;^A
.TICCB==:2                      ;^B
.TICCC==:3                      ;^C
.TICCD==:4                      ;^D
.TICCE==:5                      ;^E
.TICCF==:6                      ;^F
.TICCG==:7                      ;^G
.TICCH==:8                      ;^H
.TICCI==:9                      ;^I
.TICCJ==:10                     ;^J
.TICCK==:11                     ;^K
.TICCL==:12                     ;^L
.TICCM==:13                     ;^M
.TICCN==:14                     ;^N
.TICCO==:15                     ;^O
.TICCP==:16                     ;^P
.TICCQ==:17                     ;^Q
.TICCR==:18                     ;^R
.TICCS==:19                     ;^S
.TICCT==:20                     ;^T
.TICCU==:21                     ;^U
.TICCV==:22                     ;^V
.TICCW==:23                     ;^W
.TICCX==:24                     ;^X
.TICCY==:25                     ;^Y
.TICCZ==:26                     ;^Z
.TICES==:27                     ;ESC
.TICRB==:28                     ;RUBOUT
.TICSP==:29                     ;SPACE
.TICRF==:30                     ;CARRIER OFF
.TICTI==:31                     ;TYPEIN
.TICTO==:32                     ;TYPEOUT
```

```
RADIX 8

;CAPABILITIES

SC%CTC==:1B0                          ;CONTROL-C
SC%GTB==:1B1                          ;GETAB
SC%MMN==:1B2                          ;MAP MONITOR
SC%LOG==:1B3                          ;LOGGING FUNCTIONS
SC%MPP==:1B4                          ;MAP PRIVILEGED PAGES
SC%SDV==:1B5                          ;SPECIAL DEVICES
SC%SCT==:1B6                          ;ASSIGN TTY AS CONTROLLING FOR FORK (SCTTY)

SC%SUP==:1B9                          ;SUPERIOR ACCESS

SC%FRZ==:1B17                         ;FREEZE ON TERMINATING CONDITIONS

SC%WHL==:1B18                         ;WHEEL
SC%OPR==:1B19                         ;OPERATOR
SC%CNF==:1B20                         ;CONFIDENTIAL INFORMATION ACCESS
SC%MNT==:1B21                         ;MAINTENANCE
SC%IPC==:1B22                         ;IPCF PRIVILEGES
SC%ENQ==:1B23                         ;ENQ/DEQ PRIVILEGES
SC%NWZ==:1B24               ;TOPS20AN ;NET WIZARD PRIVILEGES (ASNSQ, ETC.)
SC%NAS==:1B25               ;TOPS20AN ;NETWORK ABSOLUTE SOCKET PRIVILEGE


;OUTMODED NAMES FOR BITS IN DIRECTORY MODE WORD - USE CD%XXX
;EQUIVALENTS

MD%FO==:CD%DIR                        ;FILES ONLY DIRECTORY
MD%SA==:CD%ANA                        ;STRING ACCOUNT ALLOWED
MD%RLM==:CD%RLM                       ;REPEAT LOGIN MESSAGE
```

```
;FDB DEFINITIONS

FB%TMP==:1B0                  ;FILE IS TEMPORARY
FB%PRM==:1B1                  ;FILE IS PERMANENT
FB%NEX==:1B2                  ;FILE DOES NOT HAVE AN EXTENSION YET
FB%DEL==:1B3                  ;FILE IS DELETED
FB%NXF==:1B4                  ;FILE IS NONEXISTENT
FB%LNG==:1B5                  ;FILE IS A LONG FILE
FB%SHT==:1B6                  ;FILE HAS COMPRESSED PAGE TABLE
FB%DIR==:1B7                  ;FILE IS A DIRECTORY FILE
FB%NOD==:1B8                  ;FILE IS NOT TO BE DUMPED BY BACKUP SYSTEM
FB%BAT==:1B9                  ;FILE HAS AT LEAST ONE BAD PAGE IN IT
FB%SDR==:1B10                 ;THIS DIRECTORY HAS SUBDIRECTORIES
FB%FCF==:17B17                ;FILE CLASS FIELD
 .FBNRM==:0                   ;NON-RMS
 .FBRMS==:1                   ;RMS FILES

.FBHDR==:0                    ;HEADER WORD
.FBCTL==:1                    ;FLAGS
.FBEXL==:2                    ;LINK TO FDB OF NEXT EXTENSION
.FBADR==:3                    ;DISK ADDRESS OF INDEX BLOCK
.FBPRT==:4                    ;PROTECTION OF THE FILE
.FBCRE==:5                    ;TIME AND DATE OF LAST WRITE
.FBUSE==:6                    ;LAST WRITER ,, AUTHOR (OBS)
.FBAUT==:6                    ;POINTER TO AUTHOR STRING
.FBGEN==:7                    ;GENERATION ,, DIR #
 FB%GEN==:777777B17           ;GENERATION NUMBER
.FBDRN==:7                    ;GENERATION ,, DIR #
 FB%DRN==:777777              ;DIR NUMBER
.FBACT==:10                   ;ACCOUNT
.FBBYV==:11                   ;RETENTION+BYTE SIZE+MODE ,, # OF PAGES
 FB%RET==:77B5                ;RETENTION COUNT
 FB%BSZ==:77B11               ;BYTE SIZE
 FB%MOD==:17B17               ;LAST OPENF MODE
 FB%PGC==:777777              ;PAGE COUNT
.FBSIZ==:12                   ;EOF POINTER
.FBCRV==:13                   ;TIME AND DATE OF CREATION OF FILE
.FBWRT==:14                   ;TIME AND DATE OF LAST USER WRITE
.FBREF==:15                   ;TIME AND DATE OF LAST NON-WRITE ACCESS
.FBCNT==:16                   ;# OF WRITES ,, # OF REFERENCES
.FBBK0==:17                   ;BACKUP WORDS (5)
.FBBK1==:20
.FBBK2==:21
.FBBK3==:22
.FBBK4==:23
.FBUSW==:24                   ;USER SETTABLE WORD
.FBGNL==:25                   ;LINK TO NEXT GENERATION FILE
.FBNAM==:26                   ;POINTER TO NAME BLOCK
.FBEXT==:27                   ;POINTER TO EXTENSION BLOCK
.FBLWR==:30                   ;POINTER TO LAST WRITER STRING

.FBLN0==:30                   ;LENGTH OF VERSION 0 FDB
.FBLN1==:31                   ;LENGTH OF VERSION 1 FDB
.FBLEN==:31                   ;LENGTH OF THE FDB
```

```
;CARD READER DEFINITIONS

.CRILC==:"\"                    ;ILLEGAL CHARACTER CODE

;A WORD IS DISTINGUISHED FROM A BYTE POINTER BY THE VALUE 5 IN BITS 0-2
;USE THESE DEFINITIONS TO TEST FOR A NUMBER AS FOLLOWS:
;       LOAD AC,NMFLG,LOC
;       CAIE AC,NUMVAL

NMFLG==:7B2
NUMVAL==:5



;DEFINITIONS FOR COMMUNICATIONS PROTOCOLS

;DEFINE THE SUPPORTED PROTOCOL TYPES

.VN20F==:0                      ;RSX20F PROTOCOL
.VNMCB==:1                      ;MCB DECNET PROTOCOL

;DEFINE BITS USED WHEN RELOADING AN -11

RM%ROM==:1B0                    ;IF SET, ACTIVATE ROM
```

```
;***********************************************
;GENERAL FIELD AND VALUE DEFINITIONS
;USED BY TOPS20AN JSYS'S
;***********************************************

;STATES OF A CONNECTION IN ARPANET NCP
; RETURNED IN B0-B3 OF GDSTS ON A NET CONNECTION
; ALSO AVAILABLE IN A GETAB, BUT THAT'S NOT THE PREFERRED WAY
; TO READ THEM, IF YOU HAVE A JFN FOR THE CONNECTION.

.NSCZD==:01                     ;CLOSED
.NSPND==:02                     ;PENDING
.NSLSN==:03                     ;LISTENING
.NSRCR==:04                     ;REQUEST FOR CONNECTION RECEIVED
.NSCW1==:05                     ;CLOSE WAIT SUB ONE (NCP CLOSE)
.NSRCS==:06                     ;REQUEST FOR CONNECTION SENT
.NSOPN==:07                     ;OPENED
.NSCSW==:10                     ;CLOSE WAIT (NCP CLOSE)
.NSDTW==:11                     ;FINAL DATA WAIT
.NSRF1==:12                     ;RFNM WAIT SUB ONE (NORMAL NCP CLOSE)
.NSCZW==:13                     ;CLOSE WAIT (PROGRAM CLOSE)
.NSRF2==:14                     ;RFNM WAIT SUB TWO (UNEXPECTED NCP CLOSE)
.NSFRE==:16                     ;FREE
```

```
;ERROR CODE DEFINITIONS

        .ERBAS==:600000         ;BASE VALUE FOR ALL ERROR CODES

DEFINE .ERCOD <

.ERR (10,LGINX1,<Invalid account identifier>)
.ERR (11,LGINX2,<Directory is "files-only" and cannot be logged in to>)
.ERR (12,LGINX3,<Internal format of directory is incorrect>)
.ERR (13,LGINX4,<Invalid password>)
.ERR (14,LGINX5,<Job is already logged in>)
.ERR (20,CRJBX1,<Invalid parameter or function bit combination>)
.ERR (21,CRJBX2,<Illegal for created job to enter MINI-EXEC>)
.ERR (22,CRJBX3,<Reserved>)
.ERR (23,CRJBX4,<Terminal is not available>)
.ERR (24,CRJBX5,<Unknown name for LOGIN>)
.ERR (25,CRJBX6,<Insufficient system resources>)
.ERR (26,CRJBX7,<Reserved>)
.ERR (35,LOUTX1,<Illegal to specify job number when logging out own job>)
.ERR (36,LOUTX2,<Invalid job number>)
.ERR (45,CACTX1,<Invalid account identifier>)
.ERR (46,CACTX2,<Job is not logged in>)
.ERR (50,EFCTX1,<WHEEL or OPERATOR capability required>)
.ERR (51,EFCTX2,<Entry cannot be longer than 64 words>)
.ERR (52,EFCTX3,<Fatal error when accessing FACT file>)
.ERR (55,GJFX1,<Desired JFN invalid>)
.ERR (56,GJFX2,<Desired JFN not available>)
.ERR (57,GJFX3,<No JFN available>)
.ERR (60,GJFX4,<Invalid character in filename>)
.ERR (61,GJFX5,<Field cannot be longer than 39 characters>)
.ERR (62,GJFX6,<Device field not in a valid position>)
.ERR (63,GJFX7,<Directory field not in a valid position>)
.ERR (64,GJFX8,<Directory terminating delimiter is not preceded by a valid
     beginning delimiter>)
.ERR (65,GJFX9,<More than one name field is not allowed>)
.ERR (66,GJFX10,<Generation number is not numeric>)
.ERR (67,GJFX11,<More than one generation number field is not allowed>)
.ERR (70,GJFX12,<More than one account field is not allowed>)
.ERR (71,GJFX13,<More than one protection field is not allowed>)
.ERR (72,GJFX14,<Invalid protection>)
.ERR (73,GJFX15,<Invalid confirmation character>)
.ERR (74,GJFX16,<No such device>)
.ERR (75,GJFX17,<No such directory name>)
.ERR (76,GJFX18,<No such filename>)
.ERR (77,GJFX19,<No such file type>)
.ERR (100,GJFX20,<No such generation number>)
.ERR (101,GJFX21,<File was expunged>)
.ERR (102,GJFX22,<Insufficient system resources (Job Storage Block full)>)
.ERR (103,GJFX23,<Directory full>)
.ERR (104,GJFX24,<File not found>)
.ERR (107,GJFX27,<File already exists (new file required)>)
.ERR (110,GJFX28,<Device is not on line>)
.ERR (111,GJFX29,<Device is not available to this job>)
.ERR (112,GJFX30,<Account is not numeric>)
.ERR (113,GJFX31,<Invalid wildcard designator>)
.ERR (114,GJFX32,<No files match this specification>)
```

```
.ERR (115,GJFX33,<Filename was not specified>)
.ERR (116,GJFX34,<Invalid character "?" in file specification>)
.ERR (117,GJFX35,<Directory access privileges required>)
.ERR (120,OPNX1,<File is already open>)
.ERR (121,OPNX2,<File does not exist>)
.ERR (122,OPNX3,<Read access required>)
.ERR (123,OPNX4,<Write access required>)
.ERR (124,OPNX5,<Execute access required>)
.ERR (125,OPNX6,<Append access required>)
.ERR (126,OPNX7,<Device already assigned to another job>)
.ERR (127,OPNX8,<Device is not on line>)
.ERR (130,OPNX9,<Invalid simultaneous access>)
.ERR (131,OPNX10,<Entire file structure full>)
.ERR (133,OPNX12,<List access required>)
.ERR (134,OPNX13,<Invalid access requested>)
.ERR (135,OPNX14,<Invalid mode requested>)
.ERR (136,OPNX15,<Read/write access required>)
.ERR (137,OPNX16,<File has bad index block>)
.ERR (140,OPNX17,<No room in job for long file page table>)
.ERR (141,OPNX18,<Unit Record Devices are not available>)
.ERR (142,OPNX19,<IMP is not up>) ;TOPS20AN
.ERR (143,OPNX20,<Host is not up>) ;TOPS20AN
.ERR (144,OPNX21,<Connection refused>) ;TOPS20AN
.ERR (145,OPNX22,<Connection byte size does not match>) ;TOPS20AN
.ERR (150,DESX1,<Invalid source/destination designator>)
.ERR (151,DESX2,<Terminal is not available to this job>)
.ERR (152,DESX3,<JFN is not assigned>)
.ERR (153,DESX4,<Invalid use of terminal designator or string pointer>)
.ERR (154,DESX5,<File is not open>)
.ERR (155,DESX6,<Device is not a terminal>)
.ERR (156,DESX7,<JFN cannot refer to output wildcard designators>)
.ERR (157,DESX8,<File is not on disk>)
.ERR (160,CLSX1,<File is not open>)
.ERR (161,CLSX2,<File cannot be closed by this process>)
.ERR (165,RJFNX1,<File is not closed>)
.ERR (166,RJFNX2,<JFN is being used to accumulate filename>)
.ERR (167,RJFNX3,<JFN is not accessible by this process>)
.ERR (170,DELFX1,<Delete access required>)
.ERR (175,SFPTX1,<File is not open>)
.ERR (176,SFPTX2,<Illegal to reset pointer for this file>)
.ERR (177,SFPTX3,<Invalid byte number>)
.ERR (200,CNDIX1,<Invalid password>)
.ERR (202,CNDIX3,<Invalid directory number>)
.ERR (204,CNDIX5,<Job is not logged in>)
.ERR (210,SFBSX1,<Illegal to change byte size for this opening of file>)
.ERR (211,SFBSX2,<Invalid byte size>)
.ERR (215,IOX1,<File is not opened for reading>)
.ERR (216,IOX2,<File is not opened for writing>)
.ERR (217,IOX3,<File is not open for random access>)
.ERR (220,IOX4,<End of file reached>)
.ERR (221,IOX5,<Device or data error>)
.ERR (222,IOX6,<Illegal to write beyond absolute end of file>)
.ERR (240,PMAPX1,<Invalid access requested>)
.ERR (241,PMAPX2,<Invalid use of PMAP>)
.ERR (245,SPACX1,<Invalid access requested>)
```

```
.ERR (250,FRKHX1,<Invalid process handle>)
.ERR (251,FRKHX2,<Illegal to manipulate a superior process>)
.ERR (252,FRKHX3,<Invalid use of multiple process handle>)
.ERR (253,FRKHX4,<Process is running>)
.ERR (255,FRKHX6,<All relative process handles in use>)
.ERR (260,SPLFX1,<Process is not inferior or equal to self>)
.ERR (261,SPLFX2,<Process is not inferior to self>)
.ERR (262,SPLFX3,<New superior process is inferior to intended inferior>)
.ERR (267,GTABX1,<Invalid table number>)
.ERR (270,GTABX2,<Invalid table index>)
.ERR (271,GTABX3,<GETAB capability required>)
.ERR (273,RUNTX1,<Invalid process handle -3 or -4>)
.ERR (275,STADX1,<WHEEL or OPERATOR capability required>)
.ERR (276,STADX2,<Invalid date or time>)
.ERR (300,ASNDX1,<Device is not assignable>)
.ERR (301,ASNDX2,<Illegal to assign this device >)
.ERR (302,ASNDX3,<No such device>)
.ERR (320,ATACX1,<Invalid job number>)
.ERR (321,ATACX2,<Job already attached>)
.ERR (322,ATACX3,<Incorrect user number>)
.ERR (323,ATACX4,<Invalid password>)
.ERR (324,ATACX5,<This job has no controlling terminal>)
.ERR (332,STDVX1,<No such device>)
.ERR (335,DEVX1,<Invalid device designator>)
.ERR (336,DEVX2,<Device already assigned to another job>)
.ERR (337,DEVX3,<Device is not on line>)
.ERR (345,MNTX1,<Internal format of directory is incorrect>)
.ERR (346,MNTX2,<Device is not on line>)
.ERR (347,MNTX3,<Device is not mountable>)
.ERR (350,TERMX1,<Invalid terminal code>)
.ERR (351,TLNKX1,<Illegal to set remote to object before object to remote>)
.ERR (352,ATIX1,<Invalid software interrupt channel number>)
.ERR (353,ATIX2,<Control-C capability required>)
.ERR (356,TLNKX2,<Link was not received within 15 seconds>)
.ERR (357,TLNKX3,<Links full>)
.ERR (360,TTYX1,<Device is not a terminal>)
.ERR (361,RSCNX1,<Overflowed rescan buffer, input string truncated>)
.ERR (362,RSCNX2,<Invalid function code>)
.ERR (363,CFRKX3,<Insufficient system resources>)
.ERR (365,KFRKX1,<Illegal to kill top level process>)
.ERR (366,KFRKX2,<Illegal to kill self>)
.ERR (367,RFRKX1,<Processes are not frozen>)
.ERR (370,HFRKX1,<Illegal to halt self with HFORK>)
.ERR (371,GFRKX1,<Invalid process handle>)
.ERR (373,GETX1,<Invalid save file format>)
.ERR (374,GETX2,<System Special Pages Table full>)
.ERR (375,TFRKX1,<Undefined function code>)
.ERR (376,TFRKX2,<Unassigned fork handle or not immediate inferior>)
.ERR (377,SFRVX1,<Invalid position in entry vector>)
.ERR (407,NOUTX1,<Radix is not in range 2 to 36 >)
.ERR (410,NOUTX2,<Column overflow>)
.ERR (411,TFRKX3,<Fork(s) not frozen>)
.ERR (414,IFIXX1,<Radix is not in range 2 to 10>)
.ERR (415,IFIXX2,<First nonspace character is not a digit>)
.ERR (416,IFIXX3,<Overflow (number is greater than 2**35 )>)
```

```
.ERR (424,GFDBX1,<Invalid displacement>)
.ERR (425,GFDBX2,<Invalid number of words>)
.ERR (426,GFDBX3,<List access required>)
.ERR (430,CFDBX1,<Invalid displacement>)
.ERR (431,CFDBX2,<Illegal to change specified bits>)
.ERR (432,CFDBX3,<Write or owner access required>)
.ERR (433,CFDBX4,<Invalid value for specified bits>)
.ERR (440,DUMPX1,<Command list error>)
.ERR (441,DUMPX2,<JFN is not open in dump mode>)
.ERR (442,DUMPX3,<Address error (too big or crosses end of memory)>)
.ERR (443,DUMPX4,<Access error (cannot read or write data in memory)>)
.ERR (450,RNAMX1,<Files are not on same device>)
.ERR (451,RNAMX2,<Destination file expunged>)
.ERR (452,RNAMX3,<Write or owner access to destination file required>)
.ERR (453,RNAMX4,<Quota exceeded in destination of rename>)
.ERR (454,BKJFX1,<Illegal to back up terminal pointer twice>)
.ERR (460,TIMEX1,<Time cannot be greater than 24 hours>)
.ERR (461,ZONEX1,<Time zone out of range>)
.ERR (462,ODTNX1,<Time zone must be USA or Greenwich>)
.ERR (464,DILFX1,<Invalid date format>)
.ERR (465,TILFX1,<Invalid time format>)
.ERR (466,DATEX1,<Year out of range>)
.ERR (467,DATEX2,<Month is not less than 12>)
.ERR (470,DATEX3,<Day of month too large>)
.ERR (471,DATEX4,<Day of week is not less than 7>)
.ERR (472,DATEX5,<Date out of range>)
.ERR (473,DATEX6,<System date and time are not set>)
.ERR (516,SMONX1,<WHEEL or OPERATOR capability required>)
.ERR (530,SACTX1,<File is not on multiple-directory device>)
.ERR (531,SACTX2,<Insufficient system resources (Job Storage Block full)>)
.ERR (532,SACTX3,<Directory requires numeric account>)
.ERR (533,SACTX4,<Write or owner access required>)
.ERR (540,GACTX1,<File is not on multiple-directory device>)
.ERR (541,GACTX2,<File expunged>)
.ERR (544,FFUFX1,<File is not open>)
.ERR (545,FFUFX2,<File is not on multiple-directory device>)
.ERR (546,FFUFX3,<No used page found>)
.ERR (555,DSMX1,<File(s) not closed>)
.ERR (560,RDDIX1,<Illegal to read directory for this device>)
.ERR (570,SIRX1,<Table address is not greater than 20>)
.ERR (600,SSAVX1,<Illegal to save files on this device>)
.ERR (601,SSAVX2,<Page count is not less than or equal to 1000>)
.ERR (610,SEVEX1,<Entry vector is not less than 1000>)
.ERR (614,WHELX1,<WHEEL or OPERATOR capability required>)
.ERR (615,CAPX1,<WHEEL or OPERATOR capability required>)
.ERR (617,PEEKX2,<Read access failure on monitor page>)
.ERR (620,CRDIX1,<WHEEL or OPERATOR capability required>)
.ERR (621,CRDIX2,<Illegal to change number of old directory>)
.ERR (622,CRDIX3,<Insufficient system resources (Job Storage Block full)>)
.ERR (623,CRDIX4,<Superior directory full>)
.ERR (624,CRDIX5,<Directory name not given>)
.ERR (626,CRDIX7,<File(s) open in directory>)
.ERR (640,GTDIX1,<WHEEL or OPERATOR capability required>)
.ERR (641,GTDIX2,<Invalid directory number>)
.ERR (650,FLINX1,<First character is not blank or numeric>)
```

```
.ERR (651,FLINX2,<Number too small>)
.ERR (652,FLINX3,<Number too large>)
.ERR (653,FLINX4,<Invalid format>)
.ERR (660,FLOTX1,<Column overflow in field 1 or 2>)
.ERR (661,FLOTX2,<Column overflow in field 3>)
.ERR (662,FLOTX3,<Invalid format specified>)
.ERR (670,HPTX1,<Undefined clock number>)
.ERR (700,FDFRX1,<Not a multiple-directory device>)
.ERR (701,FDFRX2,<Invalid directory number>)
.ERR (710,ATNX1,<Invalid receive JFN>) ;TOPS20AN
.ERR (711,ATNX2,<Receive JFN not opened for read>) ;TOPS20AN
.ERR (712,ATNX3,<Receive JFN not open>) ;TOPS20AN
.ERR (713,ATNX4,<Receive JFN is not a NET connection>) ;TOPS20AN
.ERR (714,ATNX5,<Receive JFN has been used>) ;TOPS20AN
.ERR (715,ATNX6,<Receive connection refused>) ;TOPS20AN
.ERR (716,ATNX7,<Invalid send JFN>) ;TOPS20AN
.ERR (717,ATNX8,<Send JFN not opened for write>) ;TOPS20AN
.ERR (720,ATNX9,<Send JFN not open>) ;TOPS20AN
.ERR (721,ATNX10,<Send JFN is not a NET connection>) ;TOPS20AN
.ERR (722,ATNX11,<Send JFN has been used>) ;TOPS20AN
.ERR (723,ATNX12,<Send connection refused>) ;TOPS20AN
.ERR (724,ATNX13,<Insufficient system resources (No NVT's)>) ;TOPS20AN
.ERR (727,CVHST1,<No string for that Host number>) ;TOPS20AN
.ERR (730,CVSKX1,<Invalid JFN>) ;TOPS20AN
.ERR (731,CVSKX2,<Local socket invalid in this context>) ;TOPS20AN
.ERR (732,SNDIX1,<Invalid message size>) ;TOPS20AN
.ERR (733,SNDIX2,<Insufficient system resources (No buffers available)>) ;TOPS20AN
.ERR (734,SNDIX3,<Illegal to specify NCP links 0 - 72>) ;TOPS20AN
.ERR (735,SNDIX4,<Invalid header value for this queue>) ;TOPS20AN
.ERR (736,SNDIX5,<IMP down>) ;TOPS20AN
.ERR (737,NTWZX1,<NET WIZARD capability  required>) ;TOPS20AN
.ERR (740,ASNSX1,<Insufficient system resources (All special queues in use)>) ;TOPS20AN
.ERR (741,ASNSX2,<Link(s) assigned to another special queue>) ;TOPS20AN
.ERR (742,SQX1,<Special network queue handle out of range>) ;TOPS20AN
.ERR (743,SQX2,<Special network queue not assigned>) ;TOPS20AN
.ERR (750,RNAMX5,<Destination file is not closed>)
.ERR (751,RNAMX6,<Destination file has bad page table>)
.ERR (752,RNAMX7,<Source file expunged>)
.ERR (753,RNAMX8,<Write or owner access to source file required>)
.ERR (754,RNAMX9,<Source file is nonexistent>)
.ERR (755,RNMX10,<Source file is not closed>)
.ERR (756,RNMX11,<Source file has bad page table>)
.ERR (757,RNMX12,<Illegal to rename to self>)
.ERR (760,GJFX36,<Internal format of directory is incorrect>)
.ERR (770,ILINS1,<Undefined operation code>)
.ERR (771,ILINS2,<Undefined JSYS>)
.ERR (772,ILINS3,<UUO simulation facility not available>)
.ERR (1000,CRLNX1,<Logical name is not defined>)
.ERR (1001,INLNX1,<Index is beyond end of logical name table>)
.ERR (1002,LNSTX1,<No such logical name>)
.ERR (1003,MLKBX1,<Lock facility already in use>)
.ERR (1004,MLKBX2,<Too many pages to be locked>)
.ERR (1005,MLKBX3,<Page is not available>)
.ERR (1006,MLKBX4,<Illegal to remove previous contents of user map>)
.ERR (1007,VBCX1,<Display data area not locked in core>)
```

```
.ERR (1010,RDTX1,<Invalid string pointer>)
.ERR (1011,GFKSX1,<Area too small to hold process structure>)
.ERR (1013,GTJIX1,<Invalid index>)
.ERR (1014,GTJIX2,<Invalid terminal line number>)
.ERR (1015,GTJIX3,<Invalid job number>)
.ERR (1016,IPCFX1,<Length of packet descriptor block cannot be less than 4>)
.ERR (1017,IPCFX2,<No message for this PID>)
.ERR (1020,IPCFX3,<Data too long for user's buffer>)
.ERR (1021,IPCFX4,<Receiver's PID invalid>)
.ERR (1022,IPCFX5,<Receiver's PID disabled>)
.ERR (1023,IPCFX6,<Send quota exceeded>)
.ERR (1024,IPCFX7,<Receiver quota exceeded>)
.ERR (1025,IPCFX8,<IPCF free space exhausted>)
.ERR (1026,IPCFX9,<Sender's PID invalid>)
.ERR (1027,IPCF10,<WHEEL capability required>)
.ERR (1030,IPCF11,<WHEEL or IPCF capability required>)
.ERR (1031,IPCF12,<No free PID's available>)
.ERR (1032,IPCF13,<PID quota exceeded>)
.ERR (1033,IPCF14,<No PID's available to this job>)
.ERR (1034,IPCF15,<No PID's available to this process>)
.ERR (1035,IPCF16,<Receive and message data modes do not match>)
.ERR (1036,IPCF17,<Argument block too small>)
.ERR (1037,IPCF18,<Invalid MUTIL JSYS function>)
.ERR (1040,IPCF19,<No PID for [SYSTEM] INFO>)
.ERR (1041,IPCF20,<Invalid process handle>)
.ERR (1042,IPCF21,<Invalid job number>)
.ERR (1043,IPCF22,<Invalid software interrupt channel number>)
.ERR (1044,IPCF23,<[SYSTEM] INFO already exists>)
.ERR (1045,IPCF24,<Invalid message size>)
.ERR (1046,IPCF25,<PID does not belong to this job>)
.ERR (1047,IPCF26,<PID does not belong to this process>)
.ERR (1050,IPCF27,<PID is not defined>)
.ERR (1051,IPCF28,<PID not accessible by this process>)
.ERR (1052,IPCF29,<PID already being used by another process>)
.ERR (1053,IPCF30,<Job is not logged in>)
.ERR (1054,GNJFX1,<No more files in this specification>)
.ERR (1055,ENQX1,<Invalid function>)
.ERR (1056,ENQX2,<Level number too small>)
.ERR (1057,ENQX3,<Request and lock level numbers do not match>)
.ERR (1060,ENQX4,<Number of pool and lock resources do not match>)
.ERR (1061,ENQX5,<Lock already requested>)
.ERR (1062,ENQX6,<Requested locks are not all locked>)
.ERR (1063,ENQX7,<No ENQ on this lock>)
.ERR (1064,ENQX8,<Invalid access change requested>)
.ERR (1065,ENQX9,<Invalid number of blocks specified>)
.ERR (1066,ENQX10,<Invalid argument block length>)
.ERR (1067,ENQX11,<Invalid software interrupt channel number>)
.ERR (1070,ENQX12,<Invalid number of resources requested>)
.ERR (1071,ENQX13,<Indirect or indexed byte pointer not allowed>)
.ERR (1072,ENQX14,<Invalid byte size>)
.ERR (1073,ENQX15,<ENQ/DEQ capability required>)
.ERR (1074,ENQX16,<WHEEL or OPERATOR capability required>)
.ERR (1075,ENQX17,<Invalid JFN>)
.ERR (1076,ENQX18,<Quota exceeded>)
.ERR (1077,ENQX19,<String too long>)
```

```
.ERR (1100,ENQX20,<Locked JFN cannot be closed>)
.ERR (1101,ENQX21,<Job is not logged in>)
.ERR (1102,IPCF31,<Invalid page number>)
.ERR (1103,IPCF32,<Page is not private>)
.ERR (1104,PMAPX3,<Illegal to move shared page into file>)
.ERR (1105,PMAPX4,<Illegal to move file page into process>)
.ERR (1106,PMAPX5,<Illegal to move special page into file>)
.ERR (1107,PMAPX6,<Disk quota exceeded>)
.ERR (1110,SNOPX1,<WHEEL or OPERATOR capability required>)
.ERR (1111,SNOPX2,<Invalid function>)
.ERR (1112,SNOPX3,<.SNPLC function must be first>)
.ERR (1113,SNOPX4,<Only one .SNPLC function allowed>)
.ERR (1114,SNOPX5,<Invalid page number>)
.ERR (1115,SNOPX6,<Invalid number of pages to lock>)
.ERR (1116,SNOPX7,<Illegal to define breakpoints after inserting them>)
.ERR (1117,SNOPX8,<Breakpoint is not set on instruction>)
.ERR (1120,SNOPX9,<No more breakpoints allowed>)
.ERR (1121,SNOP10,<Breakpoints already inserted>)
.ERR (1122,SNOP11,<Breakpoints not inserted>)
.ERR (1123,SNOP12,<Invalid format for program name symbol>)
.ERR (1124,SNOP13,<No such program name symbol>)
.ERR (1125,SNOP14,<No such symbol>)
.ERR (1126,SNOP15,<Not enough free pages for snooping>)
.ERR (1127,SNOP16,<Multiply defined symbol>)
.ERR (1130,IPCF33,<Invalid index into system PID table>)
.ERR (1131,SNOP17,<Breakpoint already defined>)
.ERR (1132,OPNX23,<Disk quota exceeded>)
.ERR (1133,GJFX37,<Input deleted>)
.ERR (1134,CRLNX2,<WHEEL or O +RATOR capability required>)
.ERR (1135,INLNX2,<Invalid function>)
.ERR (1136,LNSTX2,<Invalid function>)
.ERR (1137,ALCX1,<Invalid function>)
.ERR (1140,ALCX2,<WHEEL or OPERATOR capability required>)
.ERR (1141,ALCX3,<Device is not assignable>)
.ERR (1142,ALCX4,<Invalid job number>)
.ERR (1143,ALCX5,<Device already assigned to another job>)
.ERR (1144,SPLX1,<Invalid function>)
.ERR (1145,SPLX2,<Argument block too small>)
.ERR (1146,SPLX3,<Invalid device designator>)
.ERR (1147,SPLX4,<WHEEL or OPERATOR capability required>)
.ERR (1150,SPLX5,<Illegal to specify 0 as generation number for first file>)
.ERR (1151,CLSX3,<File still mapped>)
.ERR (1152,CRLNX3,<Invalid function>)
.ERR (1153,ALCX6,<Device assigned to user job, but will be given to allocator
     when released>)
.ERR (1154,CKAX1,<Argument block too smal>)
.ERR (1155,CKAX2,<Invalid directory number>)
.ERR (1156,CKAX3,<Invalid access code>)
.ERR (1157,TIMX1,<Invalid function>)
.ERR (1160,TIMX2,<Invalid process handle>)
.ERR (1161,TIMX3,<Time limit already set>)
.ERR (1162,TIMX4,<Illegal to clear time limit>)
.ERR (1163,SNOP18,<Data page is not private or copy-on-write>)
.ERR (1164,GJFX38,<File not found because output-only device was specified>)
.ERR (1165,GJFX39,<Logical name loop detected>)
.ERR (1166,CRDIX8,<Invalid directory number>)
```

```
.ERR (1167,CRDIX9,<Internal format of directory is incorrect>)
.ERR (1170,CRDI10,<Maximum directory number exceeded;  index table needs expanding>)
.ERR (1171,DELDX1,<WHEEL or OPERATOR capability required>)
.ERR (1172,DELDX2,<Invalid directory number>)
.ERR (1173,GACTX3,<Internal format of directory is incorrect>)
.ERR (1174,DIAGX1,<Invalid function>)
.ERR (1175,DIAGX2,<Device is not assigned>)
.ERR (1176,DIAGX3,<Argument block too small>)
.ERR (1177,DIAGX4,<Invalid device type>)
.ERR (1200,DIAGX5,<WHEEL, OPERATOR, or MAINTENANCE capability required>)
.ERR (1201,DIAGX6,<Invalid channel command list>)
.ERR (1202,DIAGX7,<Illegal to do I/O across page boundary>)
.ERR (1203,DIAGX8,<No such device>)
.ERR (1204,DIAGX9,<Unit does not exist>)
.ERR (1205,DIAG10,<Subunit does not exist>)
.ERR (1206,SYEX1,<Unreasonable SYSERR block size>)
.ERR (1207,SYEX2,<No buffer space available for SYSERR>)
.ERR (1210,MTOX1,<Invalid function>)
.ERR (1211,IOX7,<Insufficient system resources (Job Storage Block full)>)
.ERR (1212,IOX8,<Monitor internal error>)
.ERR (1213,MTOX5,<Invalid hardware data mode for magnetic tape>)
.ERR (1214,DUMPX5,<No-wait dump mode not supported for this device>)
.ERR (1215,DUMPX6,<Dump mode not supported for this device>)
.ERR (1216,IOX9,<Function legal for sequential write only>)
.ERR (1217,CLSX4,<Device still active>)
.ERR (1220,MTOX2,<Record size was not set before I/O was done>)
.ERR (1221,MTOX3,<Function not legal in dump mode>)
.ERR (1222,MTOX4,<Invalid record size>)
.ERR (1223,MTOX6,<Invalid magnetic tape density>)
.ERR (1224,OPNX25,<Device is write locked>)
.ERR (1225,GJFX40,<Undefined attribute in file specification>)
.ERR (1226,MTOX7,<WHEEL or OPERATOR capability required>)
.ERR (1227,LOUTX3,<WHEEL or OPERATOR capability required>)
.ERR (1230,LOUTX4,<LOG capability required>)
.ERR (1231,CAPX2,<WHEEL, OPERATOR, or MAINTENANCE capability required>)
.ERR (1232,SSAVX3,<Insufficient system resources (Job Storage Block full)>)
.ERR (1233,SSAVX4,<Directory area of EXE file is more than one page>)
.ERR (1234,TDELX1,<Table is empty>)
.ERR (1235,TADDX1,<Table is full>)
.ERR (1236,TADDX2,<Entry is already in table>)
.ERR (1237,TLUKX1,<Internal format of table is incorrect>)
.ERR (1240,IOX10,<Record is longer than user requested>)
.ERR (1241,CNDIX2,<WHEEL or OPERATOR capability required>)
.ERR (1242,CNDIX4,<Invalid job number>)
.ERR (1243,CNDIX6,<Job is not logged in>)
.ERR (1244,SJBX1,<Invalid function>)
.ERR (1245,SJBX2,<Invalid magnetic tape density>)
.ERR (1246,SJBX3,<Invalid magnetic tape data mode>)
.ERR (1247,TMONX1,<Invalid TMON function>)
.ERR (1250,SMONX2,<Invalid SMON function>)
.ERR (1251,SJBX4,<Invalid job number>)
.ERR (1252,SJBX5,<Job is not logged in>)
.ERR (1253,SJBX6,<WHEEL or OPERATOR capability required>)
.ERR (1254,GTJIX4,<No such job>)
.ERR (1255,ILINS4,<UUO simulation is disabled>)
```

```
.ERR (1256,ILINS5,<RMS facility is not available>)
.ERR (1257,COMNX1,<Invalid COMND function code>)
.ERR (1260,COMNX2,<Field too long for internal buffer>)
.ERR (1261,COMNX3,<Command too long for internal buffer>)
.ERR (1262,COMNX4,<Invalid character in input>)
.ERR (1263,PRAX1,<Invalid PRARG function code>)
.ERR (1264,PRAX2,<No room in monitor data base for argument block>)
.ERR (1265,COMNX5,<Invalid string pointer argument>)
.ERR (1266,COMNX6,<Problem in indirect file>)
.ERR (1267,COMNX7,<Error in command>)
.ERR (1270,PRAX3,<PRARG argument block too large>)
.ERR (1271,CKAX4,<File is not on disk>)
.ERR (1272,GACCX1,<Invalid job number>)
.ERR (1273,GACCX2,<No such job>)
.ERR (1274,MTOX8,<Argument block too long>)
.ERR (1275,DBRKX1,<No interrupts in progress>)
.ERR (1276,SJPRX1,<Job is not logged in>)
.ERR (1277,GJFX41,<File name must not exceed 6 characters>)
.ERR (1300,GJFX42,<File type must not exceed 3 characters>)
.ERR (1301,GACCX3,<Confidential Information Access capability required>)
.ERR (1302,TIMEX2,<Downtime cannot be more than 7 days in the future>)
.ERR (1303,DELFX2,<File cannot be expunged because it is currently open>)
.ERR (1304,DELFX3,<System scratch area depleted; file not deleted>)
.ERR (1305,DELFX4,<Directory symbol table could not be rebuilt>)
.ERR (1306,DELFX5,<Directory symbol table needs rebuilding>)
.ERR (1307,DELFX6,<Internal format of directory is incorrect>)
.ERR (1310,DELFX7,<FDB formatted incorrectly; file not deleted>)
.ERR (1311,DELFX8,<FDB not found; file not deleted>)
.ERR (1312,FRKHX7,<Process page cannot exceed 777>)
.ERR (1313,DIRX1,<Invalid directory number>)
.ERR (1314,DIRX2,<Insufficient system resources>)
.ERR (1315,DIRX3,<Internal format of directory is incorrect>)
.ERR (1316,UFPGX1,<File is not open for write>)
.ERR (1317,LNGFX1,<Page table does not exist and file not open for write>)
.ERR (1320,IPCF34,<Cannot receive into an existing page>)
.ERR (1321,COMNX8,<Number base out of range 2-10>)
.ERR (1322,MTOX9,<Output still pending>)
.ERR (1323,MTOX10,<VFU or RAM file cannot be OPENed>)
.ERR (1324,MTOX11,<Data too large for buffers>)
.ERR (1325,MTOX12,<Input error or not all data read>)
.ERR (1326,MTOX13,<Argument block too small>)
.ERR (1327,MTOX14,<Invalid software interrupt channel number>)
.ERR (1330,SAVX1,<Illegal to save files on this device>)
.ERR (1331,MTOX15,<Device does not have Direct Access (programmable) VFU>)
.ERR (1332,MTOX16,<VFU or Translation Ram file must be on disk>)
.ERR (1333,LPINX1,<Invalid unit number>)
.ERR (1334,LPINX2,<WHEEL or OPERATOR capability required>)
.ERR (1335,LPINX3,<Illegal to load RAM or VFU while device is OPEN>)
.ERR (1336,MTOX17,<Device is not on line>)
.ERR (1337,LGINX6,<No more job slots available for logging-in>)
.ERR (1340,DESX9,<Invalid operation for this device>)
.ERR (1341,ACESX1,<Argument block too small>)
.ERR (1342,ACESX2,<Insufficient system resources>)
.ERR (1343,DSKOX1,<Channel number too large>)
.ERR (1344,DSKOX2,<Unit number too large>)
```

```
.ERR (1345,MSTRX1,<Invalid function>)
.ERR (1346,MSTRX2,<WHEEL or OPERATOR capability required>)
.ERR (1347,MSTRX3,<Argument block too small>)
.ERR (1350,MSTRX4,<Insufficient system resources>)
.ERR (1351,MSTRX5,<Drive is not on-line>)
.ERR (1352,MSTRX6,<Home blocks are bad>)
.ERR (1353,MSTRX7,<Invalid structure name>)
.ERR (1354,MSTRX8,<Could not get OFN for ROOT-DIRECTORY>)
.ERR (1355,MSTRX9,<Could not MAP ROOT-DIRECTORY>)
.ERR (1356,MSTX10,<ROOT-DIRECTORY bad>)
.ERR (1357,MSTX11,<Could not initialize Index Table>)
.ERR (1360,MSTX12,<Could not OPEN Bit Table File>)
.ERR (1361,MSTX13,<Backup copy of ROOT-DIRECTORY is bad>)
.ERR (1362,MSTX14,<Invalid channel number>)
.ERR (1363,MSTX15,<Invalid unit number>)
.ERR (1364,MSTX16,<Invalid controller number>)
.ERR (1365,DSKX01,<Invalid structure number>)
.ERR (1366,DSKX02,<Bit table is being initialized>)
.ERR (1367,DSKX03,<Bit table has not been initialized>)
.ERR (1370,DSKX04,<Bit table being initialized by another job>)
.ERR (1371,GFUSX1,<Invalid function>)
.ERR (1372,GFUSX2,<Insufficient system resources>)
.ERR (1373,SFUSX1,<Invalid function>)
.ERR (1374,SFUSX2,<Insufficient system resources>)
.ERR (1375,SFUSX3,<No such user name>)
.ERR (1376,RCDIX1,<Insufficient system resources>)
.ERR (1377,RCDIX2,<Invalid directory specification>)
.ERR (1400,RCDIX3,<Invalid structure name>)
.ERR (1401,RCDIX4,<Monitor internal error>)
.ERR (1402,RCUSX1,<Insufficient system resources>)
.ERR (1403,TDELX2,<Invalid table entry location>)
.ERR (1404,TIMX5,<Invalid software interrupt channel number>)
.ERR (1405,LSTRX1,<Process has not encountered any errors>)
.ERR (1406,SWJFX1,<Illegal to swap same JFN>)
.ERR (1407,MTOX18,<Invalid software interrupt channel number>)
.ERR (1410,OPNX26,<Illegal to open a string pointer>)
.ERR (1411,DELFX9,<File is not a directory file>)
.ERR (1412,CRDIX6,<Directory file is mapped>)
.ERR (1413,COMNX9,<End of input file reached>)
.ERR (1414,STYPX1,<Invalid terminal type>)
.ERR (1415,PMAPX7,<Illegal to map file on dismounted structure>)
.ERR (1416,DSKOX3,<Invalid structure number>)
.ERR (1417,DESX10,<Structure is dismounted>)
.ERR (1420,DSKOX4,<Invalid address type specified>)
.ERR (1421,MSTX17,<All units in a structure must be of the same type>)
.ERR (1422,MSTX18,<No more units in system>)
.ERR (1423,MSTX19,<Unit is already part of a mounted structure>)
.ERR (1424,MSTX20,<Data error reading HOME blocks>)
.ERR (1425,MSTX21,<Structure is not mounted>)
.ERR (1426,MSTX22,<Illegal to change specified bits>)
.ERR (1427,CRDI11,<Invalid terminating bracket on directory>)
.ERR (1430,MSTX23,<Could not write HOME blocks>)
.ERR (1431,ACESX3,<Password is required>)
.ERR (1432,ACESX4,<Function not allowed for another job>)
.ERR (1433,ACESX5,<No function specified for ACCES>)
```

```
.ERR (1434,STRX05,<No such user name>)
.ERR (1435,ACESX6,<Directory is not accessed>)
.ERR (1436,STRX01,<Structure is not mounted>)
.ERR (1437,STRX02,<Insufficient system resources>)
.ERR (1440,IOX11,<Quota exceeded or disk full>)
.ERR (1441,IOX12,<Insufficient system resources (Swapping space full)>)
.ERR (1442,STRX03,<No such directory name>)
.ERR (1443,STRX04,<Ambiguous directory specification>)
.ERR (1444,PPNX1,<Invalid PPN>)
.ERR (1445,PPNX2,<Structure is not mounted>)
.ERR (1446,PPNX3,<Insufficient system resources>)
.ERR (1447,PPNX4,<Invalid directory number>)
.ERR (1450,SPLX6,<No directory to write spooled files into>)
.ERR (1451,CRDI12,<Structure is not mounted>)
.ERR (1452,GFUSX3,<File expunged>)
.ERR (1453,GFUSX4,<Internal format of directory is incorrect>)
.ERR (1454,RNMX13,<Insufficient system resources>)
.ERR (1455,SJBX8,<Illegal to perform this function>)

; ERROR CODES 1456-1534 ARE AVAILABLE******

.ERR (1535,TIMX6,<Time has already passed>)
.ERR (1536,TIMX7,<No space available for a clock>)
.ERR (1537,TIMX8,<User clock allocation exceeded>)
.ERR (1540,TIMX9,<No such clock entry found>)
.ERR (1541,TIMX10,<No system date and time>)

.ERR (1550,SCTX1,<Invalid function code>)
.ERR (1551,SCTX2,<Terminal already in use as controlling terminal>)
.ERR (1552,SCTX3,<Illegal to redefine the job's controlling terminal>)
.ERR (1553,SCTX4,<SC%SCT capability required>)

; Error codes 1554-1677 are available ******

.ERR (1700,SFUSX4,<File expunged>)
.ERR (1701,SFUSX5,<Write or owner access required>)
.ERR (1702,SFUSX6,<No such user name>)
.ERR (1703,GETX3,<Illegal to overlay existing pages>)
.ERR (1704,FILX01,<File is not open>)
.ERR (1705,ARGX01,<Invalid password>)
.ERR (1706,CAPX3,<WHEEL capability required>)
.ERR (1707,CAPX4,<WHEEL or IPCF capability required>)
.ERR (1711,CAPX6,<ENQ/DEQ capability required>)
.ERR (1712,CAPX7,<Confidential Information Access Capability required>)
.ERR (1713,ARGX02,<Invalid function>)
.ERR (1714,ARGX03,<Illegal to change specified bits>)
.ERR (1715,ARGX04,<Argument block too small>)
.ERR (1716,ARGX05,<Argument block too long>)
.ERR (1717,ARGX06,<Invalid page number>)
.ERR (1720,ARGX07,<Invalid job number>)
.ERR (1721,ARGX08,<No such job>)
.ERR (1722,ARGX09,<Invalid byte size>)
.ERR (1723,ARGX10,<Invalid access requested>)
.ERR (1724,ARGX11,<Invalid directory number>)
.ERR (1725,ARGX12,<Invalid process handle>)
```

```
.ERR (1726,ARGX13,<Invalid software interrupt channel number>)
.ERR (1727,MONX01,<Insufficient system resources>)
.ERR (1730,MONX02,<Insufficient system resources (JSB full)>)
.ERR (1731,MONX03,<Monitor internal error>)
.ERR (1732,MONX04,<Insufficient system resources (Swapping space full)>)
.ERR (1733,ARGX14,<Invalid account identifier>)
.ERR (1734,ARGX15,<Job is not logged in>)
.ERR (1735,FILX02,<Write or owner access required>)
.ERR (1736,FILX03,<List access required>)
.ERR (1737,DEVX4,<Device is not assignable>)
.ERR (1740,FILX04,<File is not on multiple-directory device>)
.ERR (1741,ARGX16,<Password is required>)
.ERR (1742,ARGX17,<Invalid argument block length>)
.ERR (1743,ARGX18,<Invalid structure name>)
.ERR (1744,DEVX5,<No such device>)
.ERR (1745,DIRX4,<Invalid directory specification>)
.ERR (1746,FILX05,<File expunged>)
.ERR (1747,STRX06,<No such user number>)
.ERR (1750,MSTX24,<Illegal to dismount the Public Structure>)
.ERR (1751,MSTX25,<Invalid number of swapping pages>)
.ERR (1752,MSTX26,<Invalid number of Front-End-Filesystem pages>)
.ERR (1753,LOUTX5,<Illegal to log out job 0>)
.ERR (1754,GJFX43,<More than one ;T specification is not allowed>)
.ERR (1755,MTOX19,<Invalid terminal line width>)
.ERR (1756,MTOX20,<Invalid terminal line length>)
.ERR (1757,MSTX27,<Specified unit is not a disk>)
.ERR (1760,MSTX28,<Could not initialize bit table for structure>)
.ERR (1761,MSTX29,<Could not reconstruct ROOT-DIRECTORY>)
.ERR (1763,DSKX05,<Disk assignments and deassignments are currently prohibited>)
.ERR (1764,DSKX06,<Invalid disk address>)
.ERR (1765,DSKX07,<Address cannot be deassigned because it is not assigned>)
.ERR (1766,DSKX08,<Address cannot be assigned because it is already assigned>)
.ERR (1767,COMX10,<Invalid default string>)
.ERR (1770,MSTX30,<Incorrect Bit Table counts on structure>)
.ERR (1771,LOCKX1,<Illegal to lock other than a private page>)
.ERR (1772,LOCKX2,<Requested page unavailable>)
.ERR (1773,LOCKX3,<Attempt to lock too much memory>)
.ERR (1774,ILLX01,<Illegal memory read>)
.ERR (1775,ILLX02,<Illegal memory write>)
.ERR (1776,ILLX03,<Memory data parity error >)
.ERR (1777,ILLX04,<Reference to non-existent page>)
.ERR (2000,MSTX31,<Structure already mounted>)
.ERR (2001,MSTX32,<Structure was not mounted>)
.ERR (2002,MSTX33,<Structure is unavailable for mounting>)
.ERR (2003,STDIX1,<The STDIR JSYS has been replaced by RCDIR and RCUSR>)
.ERR (2004,CNDIX7,<The CNDIR JSYS has been replaced by ACCES>)
.ERR (2005,PMCLX1,<Illegal page state or state transition>)
.ERR (2006,PMCLX2,<Requested physical page is unavailable>)
.ERR (2007,PMCLX3,<Requested physical page contains errors>)
.ERR (2010,DLFX10,<Cannot delete directory; file still mapped>)
.ERR (2011,DLFX11,<Cannot delete directory file in this manner>)
.ERR (2012,GJFX44,<Account string does not match>)
.ERR (2013,UTSTX1,<Invalid function code>)
.ERR (2014,UTSTX2,<Area of code too large to test>)
.ERR (2015,UTSTX3,<UTEST facility in use by another process>)
```

```
.ERR (2016,BOTX01,<Invalid DTE-20 number>)
.ERR (2017,BOTX02,<Invalid byte size>)
.ERR (2020,DCNX1,<Invalid network file name>)
.ERR (2021,DCNX5,<No more logical links available>)
.ERR (2022,DCNX3,<Invalid object>)
.ERR (2023,DCNX4,<Invalid task name>)
.ERR (2024,DCNX9,<Object is already defined>)
.ERR (2025,DCNX8,<Invalid network operation>)
.ERR (2026,DCNX11,<Link aborted>)
.ERR (2027,DCNX12,<String exceeds 16 bytes>)
.ER (2030,TTYX01,<Line is not active>)
.ERR (2031,BOTX03,<Invalid protocol version number>)
.ERR (2032,MONX05,<Insufficient system resources (no resident free space)>)
.ERR (2033,ARGX19,<Invalid unit number>)
.ERR (2034,IOX69,<General temporary TAPE error code>)
.ERR (2035,COMX11,<Invalid CMRTY pointer>)
.ERR (2036,COMX12,<Invalid CMBFP pointer>)
.ERR (2037,COMX13,<Invalid CMPTR pointer>)
.ERR (2040,COMX14,<Invalid CMABP pointer>)
.ERR (2041,COMX15,<Invalid default string pointer>)
.ERR (2042,COMX16,<Invalid help message pointer>)
.ERR (2043,COMX17,<Invalid byte pointer in function block>)
.ERR (2044,NPXAMB,<Ambiguous>)
.ERR (2045,NPXNSW,<Not a switch - does not begin with slash>)
.ERR (2046,NPXNOM,<Does not match switch or keyword>)
.ERR (2047,NPXNUL,<Null switch or keyword given>)
.ERR (2050,NPXINW,<Invalid guide word>)
.ERR (2051,NPXNC,<Not confirmed>)
.ERR (2052,NPXICN,<Invalid character in number>)
.ERR (2053,NPXIDT,<Invalid device terminator>)
.ERR (2054,NPXNQS,<Not a quoted string - does not begin with double quote>)
.ERR (2055,NPXNMT,<Does not match token>)
.ERR (2056,NPXNMD,<Does not match directory or user name>)
.ERR (2057,NPXCMA,<Comma not given>)
.ERR (2060,GJFX45,<Illegal to request multiple specifications for the same attribute>)
.ERR (2061,GJFX46,<Attribute value is required>)
.ERR (2062,GJFX47,<Attribute does not take a value>)
.ERR (2063,MSTX34,<Unit is write-locked>)
.ERR (2064,GJFX48,<GTJFN input buffer is empty>)
.ERR (2065,GJFX49,<Invalid attribute for this device>)
.ERR (2077,SJBX7,<Remark exceeds 39 characters>)
.ERR (2100,DELF10,<Directory still contains subdirectory>)
.ERR (2101,CRDI13,<Request exceeds superior directory working quota>)
.ERR (2102,CRDI14,<Request exceeds superior directory permanent quota>)
.ERR (2103,CRDI15,<Request exceeds superior directory subdirectory quota>)
.ERR (2104,CRDI16,<Invalid user group>)
.ERR (2105,ENACX1,<Account validation data base file not completely closed>)
.ERR (2106,ENACX2,<Cannot get a JFN for <SYSTEM>ACCOUNTS-TABLE.BIN>)
.ERR (2107,ENACX3,<Account validation data base file too long>)
.ERR (2110,ENACX4,<Cannot get an OFN for <SYSTEM>ACCOUNTS-TABLE.BIN>)
.ERR (2111,VACCX0,<Invalid account>)
.ERR (2112,VACCX1,<Account string exceeds 39 characters>)
.ERR (2113,USGX01,<Invalid USAGE entry type code>)
.ERR (2114,BOTX04,<Byte count is not positive>)
.ERR (2115,NODX01,<Node name exceeds 6 characters>)
```

```
.ERR (2116,USGX02,<Item not found in argument list>)
.ERR (2117,CRDI17,<Illegal to create non-files-only subdirectory under
     files-only directory>)
.ERR (2120,ENQX23,<Mismatched mask block lengths>)
.ERR (2121,ENQX22,<Invalid mask block length>)
.ERR (2122,DCNX2,<Interrupt message must be read first>)
.ERR (2123,ABRKX1,<Address break not available on this system>)
.ERR (2124,USGX03,<Default item not allowed>)
.ERR (2125,IPCF35,<Invalid IPCF quota>)
.ERR (2126,VACCX2,<Account has expired>)
.ERR (2127,CRDI18,<Illegal to delete logged-in directory>)
.ERR (2130,CRDI19,<Illegal to delete connected directory>)
.ERR (2132,BOTX05,<Protocol initialization failed>)
.ERR (2133,CRDI20,<WHEEL, OPERATOR, or requested capability required>)
.ERR (2134,COMX18,<Invalid character in node name>)
.ERR (2135,COMX19,<Too many characters in node name>)
.ERR (2136,CRDI21,<Working space insufficient for current allocation>)
.ERR (2137,ACESX7,<Directory is "files-only" and cannot be accessed>)
.ERR (2140,CRDI22,<Subdirectory quota insufficient for existing subdirectories>)
.ERR (2141,CRDI23,<Superior directory does not exist>)
.ERR (2142,STRX07,<Invalid user number>)
.ERR (2143,STRX08,<Invalid user name>)
.ERR (2144,CRDI24,<Invalid subdirectory quota>)
.ERR (2145,MOTX9,<Illegal to Read backwards>)
>                               ;END OF .ERCOD DEFINITION


;DEFINE THE ERROR CODE VALUES

DEFINE .ERR (N,E,S) <
        E=:.ERBAS+N
        IFG <N-.ERMAX>,<.ERMAX==:N>>

        .ERMAX==:0

.ERCOD
```

```
;THIS SECTION CONSISTS OF SPECIAL CODE TO WRITE THE ERRMES.BIN FILE
; THE CODE IS ONLY ASSEMBLED IF .ERBLD IS PREVIOUSLY
; DEFINED TO BE NON-ZERO.

IFNDEF .ERBLD,<.ERBLD==0>

IFN .ERBLD,<

.ERGO:  MOVSI 1,(GJ%FOU!GJ%SHT) ;GET A JFN ON ERROR FILE
        HRROI 2,[ASCIZ/ERRMES.BIN/]
        GTJFN
         JRST .ERER
        MOVE 2,[440000,,OF%WR]
        OPENF                   ;OPEN THE FILE FOR WRITE
         JRST .ERER
        MOVNI 3,.ERSTE-.ERTAB   ;GET LENGTH OF FILE
        MOVE 2,[POINT 36,.ERTAB]
        SOUT                    ;OUTPUT THE ERROR FILE DATA
        CLOSF                   ;CLOSE THE FILE
         JRST .ERER
        HALTF                   ;DONE

.ERER:  MOVEI 1,101             ;TYPE OUT ERROR CODE
        HRLOI 2,400000
        SETZ 3,
        ERSTR
         JFCL
         JFCL
        HALTF

LIT

DEFINE .ERR (N,E,S) <
        .ERQQ==<.-.ERTAB>*5
        .ERQQ2==N&37777
        .ERRM1 \.ERQQ2,N,.ERQQ
        ASCII \S'@\
>

DEFINE .ERRM1 (NN,N,.ERQQ)<
   IF1,<IFDEF EZ'NN,<
        PRINTX ERROR N=NN HAS ALREADY BEEN USED
   >>
   EZ'NN==1
   RELOC .ERTAB+NN
        .ERQQ
   RELOC
>

.ERTAB: .ERMAX                  ;FIRST WORD OF TABLE IS THE LENGTH
                                ; OF THE TABLE FOR ERSTR TO USE AS
                                ; A BOUNDS CHECK.
        BLOCK .ERMAX            ;LEAVE ROOM FOR POINTERS

.ERST:  .ERCOD                  ;BUILD STRINGS AND .ERTAB
```

```
.ERSTE:                         ;END OF STRINGS

        END .ERGO

>                               ;END OF IFN .ERBLD CONDITIONAL

PURGE .ERR,REL

END
```

APPENDIX B

ACTSYM.MAC

This appendix contains the complete copy of the system file ACTSYM.MAC, which defines the symbols and macros used with the USAGE monitor call. The user must include the statement

```
SEARCH ACTSYM
```

in his program to have the symbols defined in his assembly.

```
        UNIVERSAL ACTSYM - SYMBOL FILE FOR ACCOUNTING
        SUBTTL B.A. HUIZENGA/BAH/TAH - 6-JUN-77


;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED
;  OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.
;
;COPYRIGHT (C) 1976, 1977, 1978 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.



;PARAMETERS FOR USAGE ITEM DESCRIPTORS

;FIELDS IN DATA ITEM DESCRIPTOR

US%FLG==:77B5                   ;FLAGS
    US%IMM==:1B0                ; 1 - IMMEDIATE DATA ITEM
                                ; 0 - ADDRESS OF DATA ITEM
US%TYP==:77B11                  ;TYPE CODE
    .USASC==:0                  ;ASCII
    .USSIX==:1                  ;SIXBIT
    .USOCT==:2                  ;OCTAL
    .USDEC==:3                  ;DECIMAL
    .USDAT==:4                  ;DATE-TIME
    .USTAB==:5                  ;TABLE (SPECIAL FORM)
    .USVER==:6                  ;VERSION NUMBER
    .USSPC==:7                  ;SPACE FILL

US%LEN==:777B20                 ;LENGTH
US%COD==:77777B35               ;ITEM CODE

;RECORD TYPE CODES

.UTRST==:1                      ;SYSTEM RESTART
.UTSEN==:2                      ;SESSION ENTRY
.UTCKP==:3                      ;CHECKPOINT RECORD (SYSTEM RESTART)
.UTUSB==:4                      ;FIRST RECORD OF USAGE FILE (SAME AS .UTRST)
.UTTAD==:5                      ;DATE-TIME CHANGE
.UTBAT==:6                      ;BATCH PROCESSOR
.UTINP==:7                      ;INPUT SPOOLER RECORD
.UTOUT==:10                     ;OUTPUT SPOOLER RECORD
.UTFLU==:11                     ;FILE USAGE DIRECTORY RECORD
.UTDSU==:12                     ;DISK SPINDLE USAGE RECORD
.UTMNT==:13                     ;STRUCTURE MOUNT RECORD
.UTMMT==:14                     ;TAPE MOUNT RECORD
.UTDMT==:15                     ;DECtape MOUNT RECORD
.UTFCM==:16                     ;FILE COMMAND RECORD
```

ACTSYM.MAC

COMMENT ^

The format of the data to be passed to the accounting system will consist of a list of items describing the entries in a single record.

The record descriptor list will have a header containing the record type code and the record version information.

Format of a record descriptor:

```
!=======================================================!
!  DEC ver.   !  CUST ver.  !         Entry Type        !
!-------------------------------------------------------!
! Flags  !  Type  !   Length    !        Item Code      !
!-------------------------------------------------------!
!            Data or Address (-1 for default)           !
!-------------------------------------------------------!
\                           .                           \
\                           .                           \
\                           .                           \
!-------------------------------------------------------!
!                  0 (Marks end of list)                !
!=======================================================!
```

The generation of these tables will be controlled by the UITEM. macro. All known data items will have a name generated by the use of this macro. If any application dependent items are needed the UITEM. macro may be used to generate the new item. The USENT. macro may be used to generate the first word of the entry descriptor table.

Example: (Session Entry)

```
        MOVEI T1,.USENT         ;WRITE AN ENTRY
        MOVEI T2,LOGLST         ;PARAMETER BLOCK
        USAGE                   ;INVOKE MONITOR
                                ;RETURN +1
LOGLST: USENT. (.UTSEN,1,1)     ;SESSION ENTRY TYPE
        USRTM. (JBRUNT)         ;RUN-TIME
        USACT. (<-1,,ACCTSR>)   ;ACCOUNT
        USSST. (CONSTO)         ;SESSION START TIME
        USJTY. (-1)             ;BATCH T/S FLAG
        USBJN. (JBBNAM)         ;BATCH JOB NAME
        USBSN. (JBBSEQ)         ;BATCH SEQ NUMBER
        USCOM. (<-1,,JSSRM+1>)  ;SESSION REMARK
        USCCT. (CONCON)         ;CONSOLE CONNECT TIME
        USPNM. (<SIXBIT "MONITR">,US%IMM)
        USPVR. ([EXP SVNM])     ;MONITOR NAME AND VERSION
        USNM2. (<-1,,USRNAM+1>) ;USER NAME
        0                       ;END OF LIST

        ;;; End of comment
```

```
        SUBTTL UITEM. / USENT. / USAGE. DEFINITIONS

        SALL

DEFINE UITEM. (NAME,TYPE,LEN) <
    DEFINE US'NAME'. (DATA,IMMED<0>) <
        USAGE. (.US'NAME,LEN,TYPE,IMMED,<DATA>)
    >
>

DEFINE USENT. (ETYPE,DVER,CVER) <
        BYTE (9) DVER,CVER (18) ETYPE
>

DEFINE USAGE. (CODE,LENGTH,TYPE,FLAGS,DATUM) <
        FLAGS+<TYPE>B11+<^D'LENGTH>B20+CODE
        DATUM
>

DEFINE USDSK. (TABLE) <
        USAGE. (.USDST,0,.USTAB,US%IMM,<TABLE>)
>
```

```
        SUBTTL USAGE. ITEM-CODE DEFINITIONS

DEFINE USLIST <

DEFUS (JNO,0,.USDEC,4)            ;JOB NUMBER
DEFUS (TAD,1,.USDAT,14)           ;CURRENT DATE/TIME
DEFUS (TRM,2,.USASC,1)            ;TERMINAL DESIGNATOR
DEFUS (LNO,3,.USOCT,4)            ;LINE NUMBER
DEFUS (PNM,4,.USSIX,6)            ;PROGRAM NAME (CALLER)
DEFUS (PVR,5,.USVER,15)           ;PROGRAM VERSION
DEFUS (AMV,6,.USVER,15)           ;ACCOUNTING MODULE VERSION
DEFUS (NOD,7,.USSIX,6)            ;CALLER'S LOCATION
DEFUS (PPN,10,.USOCT,12)          ;PROJECT / PROGRAMMER NUMBER (TOPS10 ONLY)
DEFUS (NM1,11,.USASC,12)          ;NAME OF USER (TOPS10)
DEFUS (SNM,12,.USASC,39)          ;SYSTEM NAME
DEFUS (MVR,13,.USVER,15)          ;MONITOR VERSION NUMBER
DEFUS (MBD,14,.USDAT,14)          ;MONITOR BUILD DATE
DEFUS (MUP,15,.USDEC,18)          ;MONITOR UPTIME (IN SECONDS)
DEFUS (ACT,16,.USASC,39)          ;ACCOUNT STRING
DEFUS (LCK,17,.USDAT,14)          ;TIME OF LAST CHECKPOINT
DEFUS (RTM,20,.USDEC,9)           ;RUNTIME IN MS
DEFUS (CTI,21,.USDEC,11)          ;CORE-TIME INTEGRAL (TOPS10 ONLY)
DEFUS (SST,22,.USDAT,14)          ;SESSION START TIME
DEFUS (JTY,23,.USDEC,1)           ;JOB TYPE (BATCH / TIMESHARING)
DEFUS (BJN,24,.USSIX,6)           ;BATCH JOB NAME
DEFUS (BSN,25,.USDEC,6)           ;BATCH SEQUENCE NUMBER
DEFUS (COM,26,.USASC,39)          ;USER COMMENT
DEFUS (DKR,27,.USDEC,8)           ;DISK READS
DEFUS (DKW,30,.USDEC,8)           ;DISK WRITES
DEFUS (VTI,31,.USDEC,11)          ;VIRTUAL CORE-TIME INTEGRAL
DEFUS (EBX,32,.USDEC,9)           ;EBOX MEGACOUNTS (CYCLES * 10^6)
DEFUS (MBX,33,.USDEC,9)           ;MBOX MEGACOUNTS (CYCLES * 10^6)
DEFUS (MCL,34,.USDEC,6)           ;MONITOR CALLS
DEFUS (MCM,35,.USDEC,6)           ;MONITOR COMMANDS
DEFUS (SCL,36,.USDEC,3)           ;SCHEDULING CLASS
DEFUS (TYI,37,.USDEC,6)           ;TTY INPUT CHARACTERS
DEFUS (TYO,40,.USDEC,6)           ;TTY OUTPUT CHARACTERS
DEFUS (TYW,41,.USDEC,6)           ;TTY WAKEUPS
DEFUS (CPN,42,.USDEC,1)           ;NUMBER OF CPUS
DEFUS (CP0,43,.USDEC,4)           ;SERIAL NUMBER OF CPU0
DEFUS (CP1,44,.USDEC,4)           ;SERIAL NUMBER OF CPU1
DEFUS (CP2,45,.USDEC,4)           ;SERIAL NUMBER OF CPU2
DEFUS (CP3,46,.USDEC,4)           ;SERIAL NUMBER OF CPU3
DEFUS (CP4,47,.USDEC,4)           ;SERIAL NUMBER OF CPU4
DEFUS (CP5,50,.USDEC,4)           ;SERIAL NUMBER OF CPU5
DEFUS (RQQ,51,.USDEC,11)          ;RUN QUEUE QUOTIENT (TOPS10 ONLY)
DEFUS (NM2,52,.USASC,39)          ;NAME OF USER (TOPS20)
DEFUS (CCT,53,.USDEC,7)           ;CONSOLE CONNECT TIME (SECONDS)
DEFUS (DTL,54,.USDAT,14)          ;DATE/TIME BEFORE CHANGE (STAD)
```

```
;DISK UTILIZATION RECORD ENTRIES

DEFUS (NRF,55,.USDEC,3)         ;NUMBER OF RECORDS FOLLOWING
DEFUS (TAL,56,.USDEC,10)        ;TOTAL ALLOCATED STORAGE
DEFUS (TUS,57,.USDEC,10)        ;TOTAL STORAGE USED
DEFUS (TNF,60,.USDEC,5)         ;TOTAL NUMBER OF FILES
DEFUS (STR,61,.USASC,6)         ;STRUCTURE NAME
DEFUS (STP,62,.USDEC,1)         ;STRUCTURE TYPE CODE
DEFUS (KTP,63,.USDEC,3)         ;CONTROLLER TYPE
DEFUS (DTP,64,.USDEC,3)         ;DEVICE TYPE
DEFUS (LIQ,65,.USDEC,6)         ;LOGGED IN QUOTA
DEFUS (LOQ,66,.USDEC,6)         ;LOGGED OUT QUOTA
DEFUS (LLI,67,.USDAT,14)        ;LAST LOGGED IN DATE/TIME
DEFUS (LAT,70,.USDAT,14)        ;LAST DISK ACCOUNTING DATE/TIME
DEFUS (EXP,71,.USASC,1)         ;EXPIRED DIRECTORY (Y/N)
DEFUS (DIR,72,.USASC,39)        ;DIRECTORY NAME
DEFUS (ALC,73,.USDEC,10)        ;ALLOCATED STORAGE
DEFUS (USG,74,.USDEC,10)        ;STORAGE USED
DEFUS (FIL,75,.USDEC,5)         ;NUMBER OF FILES
DEFUS (FON,76,.USASC,1)         ;FILES ONLY INDICATOR (Y/N)

;SPOOLER INFORMATION RECORD ENTRIES

DEFUS (SRT,77,.USDEC,9)         ;SPOOLER RUNTIME
DEFUS (SCI,100,.USDEC,11)       ;CORE-TIME INTEGRAL
DEFUS (SDR,101,.USDEC,8)        ;SPOOLER DISK READS
DEFUS (SDW,102,.USDEC,8)        ;SPOOLER DISK WRITES
DEFUS (JNM,103,.USSIX,6)        ;JOB NAME
DEFUS (QNM,104,.USSIX,3)        ;QUEUE NAME
DEFUS (SDV,105,.USSIX,6)        ;PROCESSING DEVICE
DEFUS (SSN,106,.USDEC,6)        ;SEQUENCE NUMBER
DEFUS (SUN,107,.USDEC,6)        ;SPOOLER UNITS PROCESSED
DEFUS (CRT,110,.USDAT,14)       ;CREATION DATE/TIME OF REQUEST
DEFUS (DSP,111,.USSIX,6)        ;DISPOSITION
DEFUS (TXT,112,.USASC,39)       ;OPR OR SYSTEM TEXT
DEFUS (PRI,113,.USDEC,2)        ;PRIORITY
DEFUS (SNF,114,.USDEC,5)        ;NUMBER OF FILES PROCESSED
DEFUS (SCD,115,.USDAT,14)       ;SCHEDULED DATE/TIME
DEFUS (FRM,116,.USSIX,6)        ;FORMS TYPE

;DATE/TIME CHANGE RECORD ENTIRES

DEFUS (OFD,117,.USDEC,7)        ;OFFSET IN DAYS
DEFUS (OFS,120,.USDEC,7)        ;OFFSET IN SECONDS
DEFUS (ODT,121,.USDAT,14)       ;OLD DATE/TIME

>;;; END OF USLIST
```

```
;MACRO TO DEFINE ALL USAGE. ITEM CODES

DEFINE DEFUS (NAM,VAL,TYP,LEN) <
    IF1,<IFDEF .US'NAM,<
        PRINTX .US'NAM ALREADY DEFINED
    >>
        .US'NAM==:VAL
        UITEM. (NAM,TYP,LEN)
>

;EXPAND ALL DEFINITIONS

USLIST

;SPECIAL ITEM TYPE CODE DEFINITIONS

.USDSX==:7776           ;STRUCTURE/DIRECTORY INFO WORD (SPECIAL)
.USDST==:7777           ;DISK STATISTICS TABLE POINTER


        END
```

INDEX

DECSYSTEM-20
MONITOR CALLS
Reference Manual
AA-4166C-TM

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ______________________ Date ______________

Organization ______________________________

Street ______________________________

City ______________ State ______________ Zip Code or Country ______________

Please cut along this line.

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 152
MARLBOROUGH, MA
01752

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Documentation
200 Forest Street MR1-2/E37
Marlborough, Massachusetts 01752