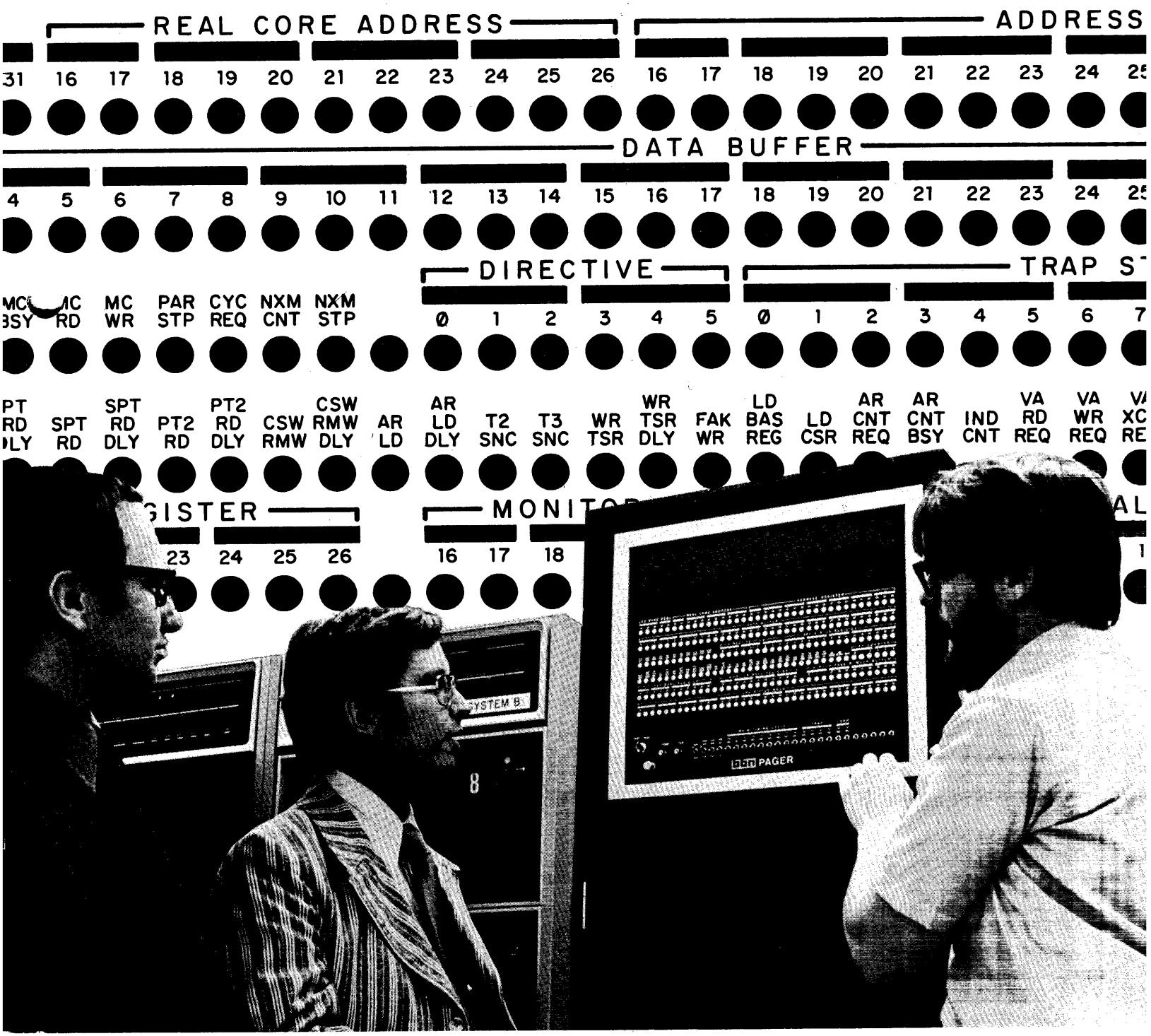
Bolt Beranek and Newman Inc.

TENEX

JSYS Manual



2nd Revision - September 1973
1st Revision - November 1971

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts
02138.

TENEX

JSYS MANUAL

A manual of TENEX monitor calls

September 1, 1973 (REVISION)

CONTENTS

INTRODUCTION TO THE TENEX JSYS MANUAL

This manual defines all of the monitor calls which exist in the TENEX System. All of these calls are by means of the JSYS instruction, detailed in the Appendix of this manual. The UUO type monitor calls (UUO's 40-77) invoke the 10/50 compatibility program which simulates the action of these UUO's in the DEC TOPS-10 monitor. See the DECsystem-10 documentation for operation of these calls.

For the sake of brevity, JSYS definitions have a very concise format. This format begins with the JSYS mnemonic and numeric definition followed by a brief description of the function of the JSYS. Next comes the line

ACCEPTS IN n:

which is followed by a list of numeric accumulators and a description of what must be in each accumulator. Next follows the JSYS mnemonic and then a number of statements of the form

```
RETURNS   +1:
          +2:
          +3:
```

which define where the JSYS code returns and under what conditions. RETURNS +1: means returns to the calling location plus 1, +2: means returns to the calling location plus 2, etc. Next there is an optional detailed description of what the JSYS does. The JSYS definition is ended with an optional list of error mnemonics. A JSYS is generally transparent to the accumulators unless a explicit mention is made of values being returned in specific accumulators.

TENEX SOURCE/DESTINATION DESIGNATORS

Several JSYS's which do such things as I/O Conversion (e.g. floating input/output, fixed input/output, directory numbers to strings and vice versa, etc.) may expect to read or write using strings in core, bytes in files, terminals, etc. In order to specify which of the above sources and destinations are being used in a single argument, a kind of universal argument called the TENEX source/destination designator has been specified. It has the formats shown below:

left half	right half	meaning
Ø	see below	a JFN (typically <155) octal
Ø	1ØØ	primary input **
Ø	1Ø1	primary output **
Ø	4xxxxx	a specific terminal number
	377777	NIL designator: infinite sink of output and infinite source of zeros on input just as a JFN for device NIL:
Ø	777777	the controlling terminal
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 44Ø7ØØ
777777	777777	universal default
5xxxxx 6xxxxx	xxxxxx xxxxxx	special cases for immediate 33 bit source number

* The JFN is the Job File Number which is a job-global handle on a file as defined in the introduction of section 2.

** These designators are legal wherever a JFN is expected.

TENEX FILE DESIGNATOR

A TENEX file designator is a subset of the TENEX source/destination designator. It includes all options except string pointers.

TENEX DEVICE DESIGNATOR

Many JSYS's dealing with special devices (Section 4) take a "TENEX device designator" as an argument. This can be either:

LH: 600000+device type number
RH: unit number, or -1 for non-units device

OR:

LH: 0
RH: 400000+a terminal number,
or -1 for controlling terminal

Terminals can thus be represented in two ways; the latter is provided for compatiblity with the TENEX source/destination designator.

The various devices are described and their type numbers are given in section 4 of this manual.

Conventions about Strings and String Pointers

A convention about passing strings and string pointers is used for most JSYS's. Strings are usually designated by the byte pointer subset of the source/destination designators. A left half of -1 is changed to 440700(8) to point to the left-most 7 bit byte of the word. JFN's and terminal designators are not legal in string pointers. The special cases of the P field equal to 5X or 6X are sometimes meaningful. This is used (for example) in the accounting JSYS's which have optional account strings or numbers and the P field 5X designates an account number.

Normally, strings being read are assumed to terminate on a null (zero) byte. Where this is not the case, another argument is necessary. This argument specifies the negative number of bytes in the string.

When not otherwise specified, a string written by a JSYS is terminated with a null byte.

When a string is read, the string pointer which was presented in an accumulator is returned in that accumulator updated to point to the character after the terminating character of the string. This means an LDB would re-read the character which terminated the string read operation. However, when a string is written which has a null character tacked onto the end, the string pointer which was presented in an accumulator is returned in that accumulator updated to point to the null character such that the next IDPB would write over the null character. When a string is written which has an arbitrary terminating character, the string pointer which was presented in an accumulator is returned in that accumulator updated to point one byte position after the terminator such that the next IDPB would append to the string without wiping out the arbitrary terminator.

UNIVERSAL FORK HANDLE

Several JSYS's take an argument called a fork handle. This is an 18 bit quantity.

```
400000 current fork
    -1 superior fork
    -2 top level fork
    -3 current fork and all its inferiors
    -4 all of the current fork's inferiors
    -5 all forks in the job
```

A fork handle may also be a number between 400001 and 400020 This is a relative handle on a fork which is only valid for a single process. This handle is in fact a dispatch into a fork handle table in the PSB of a process.

FORK/FILE HANDLE

A few JSYS's take an 18 bit argument called a fork/file handle. This handle is either a fork handle (as defined above) or a JFN. Note that string pointers and terminal identifiers cannot be specified with the handle. This is not a limitation of the handle, however, since the operations which use the fork/file handle are used for changing page maps. Such operations are not meaningful for string pointers or terminals.

TENEX DATE AND TIME STANDARDS

In TENEX, the 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 the day 1. Internal dates are number of days since November 17, 1858 at Greenwich, and thus may be a day ahead of the local date.

The internal time of day standard for TENEX is seconds since midnight (with midnight=Ø), Greenwich Mean Time (GMT).

The JSYS's which convert times and dates change from local times and dates to GMT times and dates and vice versa.

The internal format date and time are each 18-bit quantities. The date and time frequently occur in a 36-bit word as a JSYS argument or value returned; in these cases the date is always in the left half and the time in the right half.

NUMBER BASES USED IN THIS MANUAL

Except where otherwise noted, numbers used in this manual are octal, including the definitions of JSYS's. Where specific bits in words are indicated, these are numbered in decimal with the left most bit of the word called BØ and the right most bit of the word called B35.

ABBREVIATIONS

The following abbreviations are used in this manual:

BØ, B1, etc.	bit Ø, bit 1, etc. of a computer word
LH	left half (BØ-B17 of a word)
RH	right half (B18-B35 of a word)
PSI	pseudo-interrupt

Introduction Index

LOGGING FUNCTIONS

INTRODUCTION

The JSYS's defined in this section are used to initiate and delete jobs from the system as well as to append entries about these jobs to the accounting files, "<ACCOUNTS>FACT.SYS". If for any reason the primary fact file becomes fouled or indefinitely hung up, increasingly higher version numbers of the file will be used. The highest version number then becomes the primary file.

LOGIN JSYS 1

Logs a job onto the system. Useful only for logging in from an idle terminal on which a ↑C has been typed.

ACCEPTS IN 1: directory number under which one can log in for "who is on" table and file directory specification.

2: string pointer to beginning of password string in the address space of caller.

3: account number in bits 3-35 if bits Ø-2 = 5, else a string pointer to an account string in the address space of caller. If the normal terminating null byte is not seen, the string is terminated after the 8th full word is processed.

LOGIN

RETURNS +1: unsuccessful, error code in 1

+2: successful with 1 containing date and time of last login (in system internal format, see Introduction). String pointers (where used) updated in 2,3.

If successful, the user to system access words will have been set up, the job table updated, and a "TENEX-LOGIN message" appended to the accounting file. A "logged in" message is also typed on the logging terminal by this JSYS.

LOGIN ERROR MNEMONICS:

LGINX1	illegal account number or string
LGINX2	illegal directory number for logging in
LGINX4	password does not match
LGINX5	already logged in

CRJOB JSYS 2

Creates a new job (optionally logs it in)
NOT IMPLEMENTED YET

ACCEPTS IN 1: terminal number of primary input file which will be attached to the job in bits 2Ø-35 if left half is Ø and bit 18 is on, else string pointer to input file name for primary input. Name must be terminated by a legal file name terminator which includes null (same action as "space").

2: terminal number of primary output file in bits 2Ø-35 if left half is Ø and bit 18 is on, else string pointer to output file name for primary output.

3: -1 means don't log in the job, or:
directory number under which one can log in.

4: (required if 3 did not contain -1)
string pointer to beginning of password string in address space of caller

5: (required if 3 did not contain -1)
account number in bits 3-35 if bits Ø-2 = 5, else a string pointer to an account string in the address space of caller. If the normal terminating null byte is not seen, the string is terminated after the 8th full word is processed.

CRJOB

RETURNS +1: unsuccessful, error number in 1

+2: successful, job number in 3, string pointers (where used) in 1,2,4,5 returned updated.

CRJOB will assign the JSB, open the PMF, attempt to open the primary I/O files, create a top level Exec fork and start the Exec at its startup entry, and if login is invoked will do all the things the LOGIN JSYS does on success.

CRJOB ERROR MNEMONICS:

CRJBX1:	cannot open primary input file
CRJBX2:	cannot open primary output file
CRJBX3:	illegal account number or string
CRJBX4:	tty unavailable for attaching to this job
CRJBX5:	illegal directory number for logging in
CRJBX6:	no room in system
CRJBX7:	password does not match

LGOUT JSYS 3

Kills a job and appends TENEX-LOGOUT message to FACT file. Does not append this message if the job was never logged in.

ACCEPTS IN 1: the TSS job number of the job to be logged off or -1 for the running job.

LGOUT

Does not return if argument in 1 was -1

RETURNS +1: if unsuccessful, error number in 1

+2: if successful

LGOUT prints time used (both CPU and console) as well as TSS job number and current time and date on the primary output device. A similar message is typed on the logging terminal by this JSYS.

Restrictions: argument in 1 must either be -1 or must point to a job which is entered under the same user number as this job or must be executed by a job which has the LOG special privilege. "Suicide" is only permitted if the job number in 1 was -1 (i.e., if the job number presented to LGOUT is the same as that of the running job, the error return will result).

LGOUT ERROR MNEMONICS:

LOUTX1: argument in 1 points to job which violates restrictions above
LOUTX2: argument in 1 is illegal job number

CACCT JSYS 4

Changes the account number for subsequent charges to the currently entered job.

ACCEPTS IN 1: account number in bits 3-35 if bits Ø-2 = 5, else a string pointer to the new account string in the address space of caller. If the normal terminating null byte is not seen, the string is terminated after the 8th full word is processed.

CACCT

RETURNS +1: if unsuccessful, error number in 1

+2: if successful, updated string pointer in 1

Makes a "changed account" entry in the FACT file. Changes account number in JSB, resets all "charged quantities" to Ø

CACCT ERROR MNEMONICS:

CACTX1: illegal account number or string
CACTX2: not logged in

EFACT JSYS 5

Makes an arbitrary entry to FACT file.

ACCEPTS IN 1: 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)

EFACT

RETURNS +1: if unsuccessful, error number in 1

+2: if successful

EFACT returns successfully without making an entry in the FACT file if monitor flag BØ (see SMON and TMON, this section) is off.

Restrictions: this JSYS may be called only by the monitor or by processes which have the special capability LOG enabled.

EFACT ERROR MNEMONICS:

EFCTX1: LOG special capability not enabled
EFCTX2: entry too big (>=64 words)
EFCTX3: trouble with FACT file

SMON JSYS 6

Switches various monitor flags on/off (in in-core monitor and also on drum/disc image of monitor)

ACCEPTS IN 1: mask of flags to change

2: new bit values for flags indicated by 1 bits in 1

SMON

RETURNS +1: always

Requires LOG special capability.
Generates illegal instruction PSI on error conditions below.

SMON ERROR MNEMONICS:

SMONX1: LOG special capability not enabled

TMON JSYS 7

Tests to see if various monitor flags are on/off

ACCEPTS IN 1: Mask word of flags to be tested:
B0 tests fact file enabled

TMON

RETURNS +1: all flags tested are off

+2: any of the flags tested are on

Section 1 Index

JSYS's FOR REFERENCING FILES

INTRODUCTION

This section details the JSYS's for all normal references to files. All files in the system, including the system's File Directory, are normally referenced with these JSYS's. The privileged operations for directly referencing the disc, etc. are described in section 1Ø.

The JSYS's for handling subroutine files are still being specified and are not included at this writing.

File Handles

It is essential to have handles for referring to files which can be contained in a few bits and do not require extensive lookup procedures for each reference. The file name is the fundamental handle on a file, but the name fits neither criterion above. Therefore in TENEX, files are referenced by handles called JFN's (Job File Numbers). The JFN is a small number (currently Ø-155 octal) which is assigned to a file name by using the GTJFN (or GNJFN) JSYS. The same numeric handle is global to the entire job; that is, each process in a job may reference a file using the same JFN. However, the handle is not valid between jobs. That is, JFN 2 in TSS job 11 will generally be a handle on a completely different file from JFN 2 in TSS job 18.

File References in TENEX

All file operations in TENEX are initiated by acquiring a handle (JFN) on the file using the GTJFN (or GNJFN) JSYS. Some file operations such as deleting the file, renaming the file, and status queries about the file descriptors may be performed immediately after a handle is acquired for the file. Other operations, particularly any data transfer operations, require that the file be opened by performing an OPENF JSYS on the JFN. When opening a file, such things as the byte size to be used on the file for byte I/O operations and the access requested to the file are specified. Several implicit initialization operations are invoked when a file is opened which will affect subsequent references to the file. For example, the byte pointer is normally preset to the beginning of a file on opening the file such that the first sequential input operation accesses the beginning data of the file.

Sample Program

A sample program follows which acquires handles and opens both an input and an output file, then copies data from the input file to the output file in 7 bit bytes until the end of the input file is encountered. This program is highly specialized in that it will not properly copy arbitrary format (in particular randomly accessed) input files and is not anything like the general form of the EXEC's COPY command.

```
START:  RESET
        HRROI 1,[ASCIZ /
INPUT FILE: /]
        PSOUT                   ;type out string "INPUT FILE: "
        MOVSI 1,100003
        MOVE 2,[XWD 100,101]
        GTJFN                   ;GTJFN, short form, input from
                                ;and output to primary I/O device
                                ;accept existing "old" file only
        JRST START              ;start over again on failure return
        MOVEM 1,INFIL#          ;save the JFN
START1: HRROI 1,[ASCIZ /
OUTPUT FILE: /]
        PSOUT                   ;type out string "OUTPUT FILE: "
        MOVSI 1,460003
        MOVE 2,[XWD 100,101]
        GTJFN                   ;GTJFN, short form, input from
                                ;and output to primary I/O device
                                ;file is for output use, print
                                ;file type [new,old,etc.]
                                ;confirmation carriage return
                                ;is mandatory
        JRST START1             ;try again on error return
        MOVEM 1,OUTFIL#         ;save the JFN
        MOVE 1,INFIL
        MOVSI 2,7B23
        HRRI 2,201000
        OPENF                   ;open file for input, 7 bit bytes, wait
                                ;if file is busy
        JRST START              ;start all over again on failure
        MOVE 1,OUTFIL
        MOVSI 2,7B23
        HRRI 2,101000
        OPENF                   ;open file for output, 7 bit bytes,wait
                                ;if file is busy
        JRST START              ;start all over again on failure
LOOP:   MOVE 1,INFIL
        BIN                     ;input a byte from input file
        MOVE 3,2                ;save the byte in 3
        GTSTS                   ;get the status of input file
```

```
        TLNE 2,1000             ;past EOF?
        JRST DONE               ;yes, done
        MOVE 2,3                ;restore the byte to 2
        MOVE 1,OUTFIL
        BOUT                    ;output a byte to output file
        JRST LOOP
DONE:   HRRZI 1,400000
        CLZFF                   ;close all files and release all JFN's
                                ;for this process
        HALTF                   ;now quit
        END
```

File Names

File names in TENEX are composed of five identifiers. These are device, directory name, file name, extension, and version. These five items uniquely identify any file accessible to a user on the system. The device name identifies on what device in the system the file is contained. The directory name gives the directory under which the file appears. The file name and extension and version identify a particular file in the directory given by the device and directory name.

The character set from which these identifiers may be constructed is composed of the upper case letters, digits, and punctuation marks found in the range 40-137 in the ASCII set with the exception of the characters period : ; < > = ← * @ space and comma. These punctuation marks may appear in an identifier if they are preceded by a control-V. The control-V is not part of the identifier. Lower case letters may be used, but they are equivalent to the corresponding upper case letters. This is done so that all file names may be typed on upper case only terminals such as model 33 Teletype terminals. The punctuation marks listed above as exceptions are used as delimiters for various fields of the file name and while their use within a particular field may not be ambiguous from a syntactic view, a blanket restriction on their use is made for simplicity.

The general form for a file name is:

device:<directory name>name.ext;version number;T;Pprotection specification;Aaccount string

The underlined characters are real; the rest is representative.

If any of the fields of the name are omitted, the field may be supplied by the program, or from a standard set of

default values which are described with the GTJFN JSYS definition.

Recognition is done on file names in a uniform manner regardless of the source of input, or the intended use of the file. The program can control certain aspects. Whenever an alt-mode is input from memory or file, the portion of the field input prior to the alt-mode is looked up according to which field is 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 on the output file. The field terminator is output also, and recognition is done on successive fields with a null string as input, or if not possible, the fields are defaulted. If the file name cannot be uniquely determined, as many fields as possible are recognized and a bell is output signifying that more input is required. If the string input cannot possibly match any existing file name by appending more characters, an error return results.

Control-F behaves like alt-mode except recognition is not carried out past the current field. This allows the name to be recognized, for example, but not the extension.

If an alt-mode is not used, then each field is delimited as indicated in the general form above, and the name so specified must exactly match some existing file name unless the program specifies that new file names are allowed (i.e. an output file). Without alt-mode or control-F, no recognition is done but completely omitted fields will be defaulted if the program has specified default values. The complete file name is specified whenever all fields have been recognized, or a comma, space, or carriage return is input. Confirmation of file names is described with the GTJFN JSYS definition.

Editing characters are recognized while inputting file names as follows:

↑A deletes one character from the current field. If no characters remain in the current field, a bell is output.

↑W deletes the current field. If the current field is null, a bell is output.

↑X causes the file name gathering operations to be aborted, types ←←← carriage return-line feed and then restarts the file name gathering operation.

↑R retypes the entire name as specified so far and awaits further input.

If the specific call to GTJFN so permits, a star may appear in any of the directory name, file name, extension, or version fields. A star means "wild card" and will match anything. Upon completion of the GTJFN operation, the JFN returned will reference the first file found scanning in directory order which is:

In reverse by version number
In order of creation by extension
In alphabetic order by name
In numeric order by directory number.

Normally, a program accepting stars in file names will be prepared to reference all of a class of files.

File Access Protection

Because TENEX must service a diverse user community, it is essential that access to files be protected in a fairly general way. Generally, access to a file depends on two things: the kind of access desired, and the relation of the program making the access to the owner of the file. A simple protection scheme has been implemented in which the only possible relationships a program may bear to the file's owner are:

1. The directory currently connected to the job under which the program is running is the same as the owning directory ("self").

2. The directory currently connected to the job under which the program is running is in the same group as the owning directory ("group").

3. Neither 1 nor 2 ("others").

Six kinds of access are distinguished. They are:

1. read
2. write
3. execute
4. append
5. access as specified in the page table of the file
6. not used

The above six protection types and three relationships are related by 18 bits (a 3 by 6 binary matrix) in which a one indicates that a particular access is permitted for a particular relationship.

There are 36 groups available in TENEX. Both users and directories belong to groups. Relationship 2 above is satisfied if both the connected directory and the owning directory belong to the same group. Groups are assigned by administrative fiat.

File Consistency

A TENEX file may be read and written in by any number of jobs at the same time. Readers instantaneously see changes produced by writers. In most cases it is undesirable for a file to have both readers and writers or more than one writer. Therefore, each open file in the system can be opened in "thawed" mode or "frozen mode". If frozen, then all readers see an unchanging file, i.e., there can be no writers if there are readers and vice versa, and there can be at most one writer. If thawed, there can be any number of readers or writers. The mode of access is specified by the program when the file is opened (see OPENF JSYS). If the mode specified is not the same as the file's current mode, the OPENF will fail. The file's current mode is determined by the mode specified by the first opener.

Primary Input and Output Files

Each fork (process) in a TENEX job has a primary input file and a primary output file. Both are normally the controlling terminal, but can be changed to other files (with SPJFN, described in this section). The primary input and output files are referenced with designators 1ØØ and 1Ø1 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 directly (designator Ø,,777777).

Methods of Data Transfer

Most I/O in TENEX is done sequentially by bytes, as shown in the sample program above. This form of data transfer may be used with any file. A pointer maintained in the monitor is implicitly initialized when the file is opened, and advanced as data is transferred. For files on disk, there are two other possibilities. First, random access byte I/O is possible through the use of the SFPTR JSYS or RIN/ROUT (see same). Second, entire pages of data may be transferred with the PMAP JSYS.

End of File Pointer

For disc files TENEX maintains an end of file pointer, which is the highest byte address in a file to which byte output has been done. Attempting to read beyond this point in the file with byte input will produce an end-of-file condition, which can be detected with GTSTS and will produce a pseudo-interrupt if the appropriate channel is enabled (see I/O errors paragraph, next, and section 5).

Doing output by pages (PMAP JSYS) does not affect this pointer; thus it is possible for a file to contain pages of data beyond the "end of file".

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) while others are I/O device or medium errors. TENEX flags such error conditions by setting error bits in the JFN status word and by initiating pseudo-interrupt requests on the appropriate channels (see section 5). If the fork in which an I/O error occurs is not prepared to handle the pseudo-interrupt request, it is changed into a fork terminating condition with the expectation that the fork's immediate superior will handle the error condition. The TENEX EXEC is prepared to detect and diagnose I/O errors; so, a fork running directly beneath the EXEC need not do its own I/O error handling unless it chooses to do something special on I/O errors.

I/O errors can occur while a fork is executing ordinary machine instructions as well as JSYS's. For example, if a PMAP operation is done which maps a page of a file into a page of a fork, the file I/O transfer will not actually occur until a reference is made by the fork to that particular page of the fork. 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 which is assigned to another job (as a controlling terminal or with ASND) normally produces an error, but is legal if the process has the WHEEL special capability enabled.

Note: page numbers 9-14 are reserved for expansion of this introduction.

GTJFN JSYS 2Ø

Assigns a JFN to a file name for this job. Accepts a string (from a string pointer, a file, or both) of the form:

device:<directory>name.extension;version;T;Pprot;Aacct

The maximum number of JFN's a job may have in use is currently over 1ØØ.

Two types of calls to GTJFN, short and long, are possible. They are distinguished by bit 17 of AC1.

Short GTJFN call:

In the short call, the file name can be taken from a string or a file (here "file" includes terminal, primary input file, etc.), but not both. The device, directory, name, extension, etc. are defaulted to the standard system values given below, if not specified by the input string or file.

ACCEPTS IN 1: BØ-B8 and B11,B12: flags - see below
B16: ON to cause 2 to be taken as INJFN,,OUTJFN
OFF to cause 2 to be taken as string pointer
B17: ON.
RH: default version (usually Ø; see below)

2: source designator for file name, i.e. string pointer or XWD input JFN, output JFN, according to B16 of 1.
To omit either JFN, give 377777.
(the output JFN is used for printing the rest of recognized names).

GTJFN

RETURNS +1: failure, error code in 1

+2: success, JFN assigned in RH 1,
flags in LH 1 as described below,
updated string pointer in 2 (if pertinent)

Long GTJFN call:

In the long GTJFN call, input may be from both a string and a file. If both are given, the string is used, then the file is used if more characters are needed. Also, in the long call the caller has the option of specifying non-standard default values for device, directory, name, extension, etc, and of specifying the JFN he wishes assigned.

ACCEPTS IN 1: RH: pointer to a table starting at E in the caller's address space.
B17: OFF

2: main string pointer. Ø if none.

at E: XWD flags (see below), default version number
E+1: XWD input JFN, output JFN. 377777 for none.
E+2: DSP device
E+3: DSP directory
E+4: DSP name
E+5: DSP extension
E+6: DSP protection (numeric)
E+7: DSP account (string or numeric)
E+1Ø: desired JFN if flag B9 on

DSP means default string pointer. If any of these is Ø, the system default for that field is assumed (see below). Numeric values (E+6, E+7) are represented by the form 5B2+N.

GTJFN

RETURNS +1: failure, error code in 1

+2: successful, assigned JFN in RH 1,
updated string pointer in 2 (if pertinent)
flags in LH 1 as described below.

Flags accepted by GTJFN (in 1 in short call, E in long call):

B0 for output use
(affects version number defaulting. see below)
B1 new file only
B2 old file only
B3 print the appropriate one of the following messages:
[NEW FILE]
[NEW VERSION]
[OLD VERSION]
[OK] non-directory device, B4 off
[CONFIRM] non-directory device, B4 on
B4 confirmation required
B5 temporary
B6 retype whole name when complete
B7 no access by other forks
B8 ignore "deleted" bit in descriptor block
(useful for undeleting files)
B9&B10 00,01 means do not use E+10
10 try to assign JFN at E+10, error on failure
11 try to assign JFN at E+10, assign some other JFN on failure
B11 Accept input (first type) file group descriptor as defined below. If any *'s in string or in defaulted fields, assign JFN to first file in group. Verify that non-* fields are old or new per B1 and B2 in the same manner whether or not *'s are present.
B12 Accept output (second type) file group descriptor as defined below. Do not assign JFN to file even if no *'s input.
B13 Return flags in LH

On success GTJFN returns the following flags in 1 if flag B11, B12 or B13 were on in the call.

B0 * for device field
B1 * for unit
B2 * for directory
B3 * for name
B4 * for extension
B5 * for version
B6 use highest version
B7 use next higher version
B8 use lowest version
B9 protection given
B10 account given
B11 ;T given
B12 complement of B8 in call
B13 always 0

Flags BØ-B5 are set whether the * came from input or from the default string. Flags B6-B8 apply when the version is input as or defaulted to Ø, -1, and -2 respectively. Bits 12 and 13 control inclusion of deleted and undeleted files by subsequent GNJFN's. On means that class will be skipped, off means that class will be included, e.g. 1Ø means skip deleted files, include undeleted files.

System Default Values

The following file designator field values are used when the input does not specify the field and the GTJFN call does not give a default:

device	DSK:
directory	connected directory if device is DSK:
name	none
extension	null
version	see below
protection	directory's default protection
account	LOGIN account

Version Number Defaulting

Normal version number defaulting is specified by Ø in RH of 1 (short call) or E (long call) and is as follows: If "for output use" flag (BØ) is off in the call, the highest existing version is used; if this flag is on, the next higher version number is used. Other default version specifications are: a number from 1 to 131Ø71, to specify defaulting to that version; -1, to indicate the next higher version; -2, to indicate lowest existing version, -3 = *.

Terminating Character

GTJFN always reads the terminating character after the file name string. To get that character, use BKJFN then BIN.

Confirmations are:

↑I, ↑J, ↑L, ↑M, ESC, EOL, SPACE, COMMA, @, ←

File Group Descriptors

A "file group descriptor" is a TENEX file name with one or more of the fields replaced with *'s. File group descriptors are used in conjunction with the GTJFN and GNJFN JSYS's to obtain "indexable file handles" which permit programs to input "file group descriptors" and to loop over all of the designated files.

There are two types of file group descriptors, which are similar in syntax but different in meaning. The first allows looping over an existing group of files in the directory and is generally used for input files. The second, less common, type is used for output files in two-argument operations such as the Exec COPY command. In this case *'s mean "use the current value of corresponding field from the input descriptor" rather than "index this field over all existing values". The second type provides the capability of processing a group of files to yield another group of files.

Internally, an "indexable file handle" is a 36-bit quantity consisting of flags (left half) and a JFN (right half). The use of B11 and B12 in the flag word of the GTJFN JSYS allows it to input file group descriptors of both types, returning an indexable file handle to the caller. The GNJFN JSYS steps an indexable file handle representing a file group descriptor of the first type to the next file in sequence. For file group descriptors of the second type, the user program will generate successive file names.

In the first type of file group descriptor, generally used for input files, GTJFN determines that at least one value exists for each field for which an "*" is given and assigns a JFN to the first file in the group.

In the second type of file group descriptor, generally used for output files, "*" means "copy field from input file name". GTJFN collects a string and stores the non-* parts of it in the JSB for access by the JFNS JSYS, but does not assign the JFN to a specific file. Non-* fields preceding the first * are verified for existence if B2 of the flag word is on in the call.

For both cases, it is possible to specify "*" for default name, extension, version, and directory by having the appropriate default string be an "*". Also, if the version number is defaulted, the actual version will be determined independently for each file in the group. Hence in such cases nothing is printed for version after altmode termination, and appropriate flags are returned to the caller for communication to GNJFN. More specifically, after altmode, GTJFN does not print ";" and a default version if either: B11 was on in call and an * has been entered and default version given in call is Ø, -1, or -2, or B12 was on in call and default version given in call is Ø, -1, or -2. The confirmation message will be either [CONFIRM] or [OK] if an * has been input since some of the files in the group could be old and others new. More specifically, the confirmation printout is changed to [OK] or [CONFIRM] if either: B11 was on in call and at least one * has been entered, or B12 was on in call.

GTJFN ERROR MNEMONICS:

GJFX1: illegal JFN
GJFX2: not an available JFN
GJFX3: no JFN available
GJFX4: illegal character in file name
GJFX5: input field too large (>39 chars)
GJFX6: too many device fields or device not specified first
GJFX7: too many directory fields or directory after name
GJFX8: > not following <
GJFX9: too many name fields
GJFX1Ø: non-numeric version
GJFX11: version specified twice
GJFX12: too many account fields
GJFX13: too many protection fields
GJFX14: non-numeric or negative or unrecognized protection
GJFX15: improper confirmation character
GJFX16: no such device
GJFX17: no such directory
GJFX18: no such name
GJFX19: no such extension
GJFX2Ø: no such version
GJFX21: no such file (it was expunged from the directory during the execution of the GTJFN)
GJFX22: no room in JSB
GJFX23: no room in directory
GJFX24: no new files
GJFX25: non-null name used with non-directory device
GJFX26: non-null extension used with non-directory device
GJFX27: old file not allowed (bit 1 of flags set & old file specified)
GJFX28: mountable device not mounted
GJFX29: device not available to this job
GJFX3Ø: string account not allowed
GJFX31: illegal *
GJFX32: * for name in empty directory
GJFX33: null string not allowed for name
GJFX34: un-quoted "?" used in a file-name
GJFX35: read access not allowed for requested directory
GJFX36: system error-requested directory cannot be read.

GNJFN JSYS 17

Assigns JFN to next file in group as determined by its current association and flags transmitted by user from GTJFN to GNJFN. Scans only in internal directory order.

ACCEPTS IN 1: Indexable file handle (as returned by GTJFN)
LH: flags - see GTJFN description
RH: JFN

GNJFN

RETURNS +1: Error, or no more files, JFN released if no more files. Occurs on first call if no * flags on in LH of 1.

+2: Same JFN assigned to next file, previous file closed if open.

Bits returned in LH of 1 indicate which fields were changed as follows:

14 - directory changed
15 - name changed
16 - extension changed

Note that GNJFN makes use of the flags returned in LH of 1 by GTJFN to specify which fields are being indexed and what the default version is. The flags returned by GNJFN should not be used as an argument to GNJFN. That is, do not update the GTJFN flags in the user program with those from GNJFN.

Generates illegal instruction PSI on error conditions below.

GNJFN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
GNJFX1:	cannot close the file
OPNX1:	file open

OPENF JSYS 21

Opens a specified file

ACCEPTS IN 1: the JFN

2: BØ-B5: BYTE SIZE (max 36)
B6-B9: FILE DATA MODE
Ø: normal mode (dump I/O illegal)
17: dump mode (device dependent; cannot do byte I/O; byte size ignored)

Other modes are used in some devices. See, for example, NETWORK later in this section.

B1Ø-17: Device dependent (see sec. 4)
B18: Not used
B19: read
B2Ø: write
B21: execute
B22: append
B23: access as specified by the page table of the file
B24: protected entry only
NOT IMPLEMENTED YET
B25: thawed
B26: always wait if file is busy *
B27: do not change accessed dates
B28: never wait, give error if file is busy *

* B26 and B28 both off, normal device-dependent default on waiting: e.g., wait for LPT:, not for DSK:.

OPENF

RETURNS +1: unsuccessful, error # in 1

+2: successful

OPENF ERROR MNEMONICS:

OPNX1: already open
OPNX2: file doesn't exist
OPNX3: read access not allowed
OPNX4: write access not allowed
OPNX5: execute access not allowed
OPNX6: append access not allowed
OPNX7: device assigned to another job

OPNX8:	device is not mounted or is off-line
OPNX9:	file busy
OPNX1Ø:	no room
OPNX12:	list access not allowed
OPNX13:	illegal access
OPNX14:	illegal mode
OPNX15:	non-read/write access not allowed
OPNX16:	file has bad index block
OPNX2Ø:	foreign host dead (NETWORK only)
OPNX21:	rejected by foreign host (NETWORK only)
OPNX22:	byte size does not match RFC specifications (NETWORK. receive connections only)
DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
SFBSX2:	illegal byte size

CLOSF JSYS 22

Closes a specific file or all files

ACCEPTS IN 1: the JFN (if BØ is on, do not release the JFN if BØ is off, do release the JFN)
or -1=> close all files and all JFN's at or below this fork (with the exception of the primary I/O files and files which cannot be closed by the current fork), but do not release the JFN's.

CLOSF

RETURNS +1: unsuccessful, error code in 1

+2: successful

CLOSF ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
CLSX1:	not open
CLSX2:	not closable by this fork all output errors may occur

CLZFF JSYS 34

CLoZe Fork's Files. Closes all files and/or releases all JFN's at and/or below a specified fork

ACCEPTS IN 1: RH: a fork handle
- BØ on, don't close files of specified fork's inferiors
- B1 on, don't close specified fork's files
- B2 on, don't release JFN's
- B3 on, don't close any files, only release non-open JFN's
- B4 on, unrestrict files opened with restricted access for specified fork (fork must be self or inferior)
- B5 on, release specified JFN's even if a fork is sharing pages with the file

CLZFF

RETURNS +1: always, having done nothing if the call was in any way illegal

RLJFN JSYS 23

Releases JFN's

ACCEPTS IN 1: the JFN or -1=> all JFN's which do not specify open files

RLJFN

RETURNS +1: unsuccessful, error code in 1

+2: successful

RLJFN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
RJFNX1:	specified file is open hence JFN cannot be released
RJFNX2:	JFN is being used to accumulate a name hence cannot be released
RJFNX3:	JFN not accessible from this fork

RESET JSYS 147

Resets and initializes many things about the current fork; should be used at the beginning of all programs.

RESET

RETURNS +1: always

In detail, RESET does the following:

Closes all files and releases all JFN's at or below this fork

Kills all inferior forks

Clears fork's PSI system

Clears compatibility entry vector

Sets the following fields of the controlling terminal's mode word (rest of word unchanged) (see section 4B):
- Wakeup on every character
- Echo: immediate or deferred
- Lower to upper case input conversion off
- Lower case output: do not indicate by %

Control character output control:
- send CR, LF, ↑G
- simulate EOL, VTAB
- simulate or send FF and TAB per B1 and B2 of mode word
- indicate all others with ↑X

Sets tab stops every 8 columns

JFN STATUS WORD FORMAT

The two JSYS definitions which follow manipulate the JFN status word whose format is shown below

BØ: file is open
B1: file is ok to read (ie it is open for reading)
B2: file is ok to write
B3: file is ok to execute
B4: ok to reset byte pointer (not append)
B5: ok to access as specified by page table
B6: (not used)
B7: file is a long file
B8: set if last read was past end of file
B9: bytes read may be in error
B1Ø: name is associated with JFN
B11: an * was typed in one of the file name fields
B12: JFN being assigned
B13: file data errors are terminating conditions even if error interrupt is armed
B16: ok to change byte size
B17: file is restricted to some fork
B32-B35: data mode, see OPENF

GTSTS JSYS 24

Reads the status of a file

ACCEPTS IN 1: the JFN

GTSTS

RETURNS +1: always, status in 2. If JFN is in any way illegal, B1Ø of 2 will be Ø

STSTS JSYS 25

Writes the status of a file.

ACCEPTS IN 1: the JFN

2: the status word (only bits 9, 13, 17 can be changed)

STSTS

RETURNS +1: unsuccessful, error code in 1

+2: successful

STSTS ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal

DELF JSYS 26

Deletes a specified file on a directory device (DSK: or DTA:, currently). On DSK:, the file is not really deleted, but marked for later deletion by the backup system or with DELDF. Does nothing if JFN is assigned to a non-directory device.

ACCEPTS IN 1: the JFN of the file to be deleted, unless BØ on in CALL AC1.

DELF

RETURNS +1: unsuccessful, error code in 1

+2: successful, file closed (if open), JFN released, unless BØ on in CALL AC1.

DELF ERROR MNEMONICS:

DESX1: illegal value for designator
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DELFX1: attempt to delete a file for which this user does not have write permission

DELNF JSYS 317

DELete all but N versions of a File.

ACCEPTS IN 1: A JFN of a disk file

2: the number of versions to keep

Starting with the specified file version, and going down, the first N versions are retained and the next are deleted.

DELNF

RETURNS +1: unsuccessful, error number in 1

+2: successful, with the negative of the number of files deleted in 2

Like DELF, DELNF marks the files for deletion. They are expunged later by DELDF or LGOUT.

SFPTR JSYS 27

Sets the pointer in a file.

ACCEPTS IN 1: the JFN of the file

2: the byte number to which the pointer is to be set (-1 means to current end of file)

If the pointer is set beyond the end of file, a later BIN/SIN will return a zero byte with an end of file indication. (File status bit, EOF PSI if enabled). On the other hand, if an output is done (BOUT/SOUT), the byte is stored at the position specified and the file length is updated to contain the new information.

SFPTR

RETURNS +1: unsuccessful, error number in 1

+2: successful

SFPTR ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
SFPTX1:	specified file is not open
SFPTX2:	illegal to reset pointer for this file
SFPTX3:	illegal byte number for pointer reset (eg <-1)

JFNS JSYS 3Ø

JFN to string conversion

ACCEPTS IN 1: TENEX destination designator, an ASCIZ string is written

2: RH: JFN
LH: Ø or flags returned in LH 1 by GTJFN

3: output specification described in detail below, Ø for normal which is:
DEV:<DIR>NAME.EXT;VERS;T with defaulted fields not printed and account and protection also printed if B9 or B1Ø of 2 respectively on

The detailed description of each bit of this argument follows:

BØ-2	device output control
B3-5	directory output control
B6-8	name output control (2 is illegal)
B9-11	extension output control (2 is illegal)
B12-14	version output control
B15-17	protection output control
B18-2Ø	account output control
B21	print ;T if appropriate
B22	print size in pages
B23	print creation date
B24	print write date
B25	print read date
B26-3Ø	unused
B31-35	format control code

format control bits

If B31-35 = Ø, put nothing between fields

B35 on - punctuate device field through ;T field
B34 on - tab before fields which might print,
i.e., those that have format specification 1 or 2
(except for the first such field)
B33 on - tab before all (printed) fields except
first field
B32 on - punctuate size & date fields

punctuation used
punctuation is underscored

device:
<directory>
name
.ext
;version
;Aaccount
;Pprotection
;T
,size
,creation date,write date,read date

output control specification

Ø don't print
1 always print
2 suppress if system default

JFNS

RETURNS +1: always, updated string pointer (if pertinent) in 1

If the JFN given to JFNS is associated with a file and LH 2 is Ø, JFNS will print the specific file's name, version, etc. For instance, it does not print "-3" for version even if version are being scanned, and of course it does not print *'s.

If the JFN was obtained with the output file group descriptor flag (flag B12) case of GTJFN, it will be assigned but not associated with a file. In this case if LH 2 is Ø, JFNS will return nulls for non-existent fields and the proper values for existent fields.

In all cases if JFNS is supplied with flag bits in LH 2, it will obey them, printing *'s for fields whose flag is on, and printing version Ø, -1, and -2 for B6, B7, and B8 respectively.

Generates illegal instruction PSI on error conditions below.

JFNS ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: not open

FFFFP JSYS 31

Finds the First Free File Page

ACCEPTS IN 1: JFN

FFFFP

RETURNS +1: with JFN in left half of 1, page number in right half of 1, OR -1 in 1 if no free pages

Generates illegal instruction PSI on error condtions below

FFFFP ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
DESX5:	not open

FFUFP JSYS 211

Finds the First Used File Page at or beyond a specified page number.

ACCEPTS IN 1: LH: JFN
RH: starting page number

FFUFP

RETURNS +1: failure, error number in 1

+2: success, page number in RH 1,
LH 1 preserved.

FFUFP ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
FFUFX1:	not open
FFUFX2:	file not on multiple directory device
FFUFX3:	no used page found

CPRTF JSYS 33

Changes protection of a file
NOT IMPLEMENTED YET USE CHFDB

ACCEPTS IN 1: JFN of file whose protection is to be changed

2: new protection:
XWD 500000,18 bit protection,
Or, later, a string pointer to a protection name.

CPRTF

RETURNS +1: unsuccessful, error code in 1

+2: successful

CPRTF ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this jfn
DESX4: terminal designator or string pointer not legal
CPRTX1: this file's current protection does not permit you to change its protection

RNAMF JSYS 35

Renames an existing file to another name

ACCEPTS IN 1: JFN of existing file to be renamed

2: JFN of renamed file name (note if the renamed file name is that of an existing file, the existing file will get marked as deleted in such a way that it cannot be undeleted!)

RNAMF

RETURNS +1: unsuccessful, error code in 1

+2: successful, JFN in 2 now talking to same file, new name, JFN in 1 gets released.

RNAMF ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
OPNX1:	already open, unopened file names must be used
RNAMX1:	attempt to rename files which are not on same device
RNAMX2:	destination non-existent
RNAMX3:	no access to destination
RNAMX4:	no space to do rename
RNAMX5:	destination busy
RNAMX6:	destination has bad page table
RNAMX7:	source non-existent
RNAMX8:	no access to source
RNAMX9:	source is empty
RNAMX1Ø:	source busy
RNAMX11:	source has bad page table
RNAMX12:	rename to self

SIZEF JSYS 36

Gets length of an existing file

ACCEPTS IN 1: JFN of existing file

SIZEF

RETURNS +1: unsuccessful, error code in 1
+2: successful, byte count which referenced the last byte written into the file in 2
# of pages (512 words) in file in 3

NOTE: Use GTFDB to get the byte size in which the file was written. For a holey file the byte count in 2 does not necessarily reflect the file's size.

SIZEF ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal

GACTF JSYS 37

Gets account number or string to which existing file is being charged

ACCEPTS IN 1: JFN

2: pointer to E where account string (if any) will get stored.

GACTF

RETURNS +1: illegal, error # in 1

+2: string returned starting at E, updated string pointer in 2

+3: number returned in 2, with BØ and B2 on for consistency with SACTF

Related JSYS: SACTF, see index

GACTF ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
GACTX1: not multiple directory device
GACTX2: file not found

STDIR JSYS 40

Translates string to directory number

ACCEPTS IN 1: if positive use entire string literally for exact match
if negative try to recognize
if B17 is 1, use specified device
otherwise use default device

2: string pointer

3: device designator if B17 of AC1 on

STDIR

RETURNS +1: no match

+2: ambiguous

+3: unique match, directory number returned in right half of 1.
left half of 1, B0 off says user may log in under or connect to this directory, B0 on says name may only be used for directory connection.
B1 off says user must specify account as number, B1 on says user may use alphanumeric account string.
B2 off says print login message only if newer than date and time of last LOGIN, B2 on says repeat LOGIN message at every LOGIN.
Appends remainder of string (if any) to original string if recognition was invoked and updated string pointer is returned in 2.

DIRST JSYS 41

Translates directory number to string

ACCEPTS IN 1: TENEX destination designator

2: directory number

DIRST

RETURNS +1: unsuccessful, number not in use

+2: match found, string written to destination, updated string pointer (if pertinent) in 1

Generates illegal instruction PSI on error condtions below

DIRST ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX5: not open

BKJFN JSYS 42

Backs up the pointer in a file by one byte

ACCEPTS IN 1: TENEX source/destination designator

BKJFN

RETURNS +1: unsuccessful, error # in 1

+2: successful, updated string pointer (if pertinent) in 1 - (this actually will decrement a string pointer)

BKJFN cannot be used to back up more than one character for terminals.

NOTE: Not implemented for DECtape files.

BKJFN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	not open
BKJFX1:	illegal attempt to back up terminal pointer twice
SFPTX2:	illegal to reset pointer for this file
SFPTX3:	attempt to reset pointer to before beginning of file

RFPTR JSYS 43

Reads position of pointer in file

ACCEPTS IN 1: JFN

RFPTR

RETURNS +1: unsuccessful, error # in 1

+2: byte number in 2

RFPTR ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
DESX5:	not open

CNDIR JSYS 44

Connects the job to a file directory.

ACCEPTS IN 1: a directory # to which to connect the job. If the sign bit is set, this will just check the password without connecting to the directory.

2: a string pointer to a password string (optional, not needed if directory is that under which job is logged in, or if directory does not have password string)

CNDIR

RETURNS +1: unsuccessful, error # in 1

+2: successful, string pointer in 2 not updated

NOTE: previous file openings remain valid after a CNDIR

CNDIR ERROR MNEMONICS:

CNDIX1: incorrect password
CNDIX3: invalid directory #
CNDIX4: logged in
CNDIX5: not logged in

RFBSZ JSYS 45

Reads byte size for a specific opening of a file

ACCEPTS IN 1: JFN

RFBSZ

RETURNS +1: with byte size right justified in 2

Generates illegal instruction PSI on error conditions below

RFBSZ ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: not open

SFBSZ JSYS 46

Resets byte size for a specific opening of a file

ACCEPTS IN 1: JFN

2: byte size

SFBSZ

RETURNS +1: always

Generates illegal instruction PSI on error conditions below

SFBSZ ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: not open
SFBSX1: illegal to change byte size for this jfn
SFBSX2: illegal byte size

SWJFN JSYS 47

Swaps the association of two JFN's by literally exchanging all information cells of each JFN

ACCEPTS IN 1: JFN

2: another JFN

SWJFN

RETURNS +1: always

Generates illegal instruction PSI on error conditions below.

SWJFN ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal

PBIN JSYS 73

Inputs the next sequential byte from the primary input file.

PBIN

RETURNS +1: with byte right justified in 1

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual).

PBIN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	file not open
IOX1:	illegal read
IOX4:	end of file
IOX5:	data error

PBOUT JSYS 74

Outputs a byte (sequentially) to primary output file.

ACCEPTS IN 1: the byte (right justified)

PBOUT

RETURNS +1: always

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual)

PBOUT ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	file not open
IOX2:	illegal write
IOX5:	data error
IOX6:	attempt to write beyond abs max file length

PSOUT JSYS 76

Outputs a string (sequentially) to primary output file.

ACCEPTS IN 1: a string pointer to a string in the caller's address space, terminated with a null character

PSOUT

RETURNS +1: always, with updated string pointer in 1

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual.)

PSOUT ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	file not open
IOX2:	illegal write
IOX5:	data error
IOX6:	attempt to write beyond abs. max. file length

ESOUT JSYS 313

Error String Output

Used to report an error in input from the primary input stream, to cause re-synchronization of the input interaction.

ACCEPTS IN 1: a string pointer to a string in caller's address space, terminated by a null character

ESOUT

RETURNS +1: always, with updated string pointer in 1

Can cause several pseudo-interrupts or fork termination on certain file conditions (see OPENF)

ESOUT first waits for the primary output buffer to empty, then outputs a carriage return, linefeed, question mark to the primary output. Next it clears the primary input buffer, and finally outputs the argument (error string) to the primary output.

This mechanism is convenient for re-synchronizing communication with a user who made a typing error, and (possibly) continued to type ahead. It also standardizes the format of error messages.

BIN JSYS 5Ø

Does sequential byte input from a source

ACCEPTS IN 1: TENEX source designator

BIN

RETURNS +1: with the byte right justified in 2

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual)

BIN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	file not open
IOX1:	illegal read
IOX4:	end of file
IOX5:	data error

BOUT JSYS 51

Does sequential byte output to a destination

ACCEPTS IN 1: TENEX destination designator

2: the byte right justified

BOUT

RETURNS +1: always

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual)

BOUT ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	file not open
IOX2:	illegal write
IOX5:	data error
IOX6:	attempt to write beyond abs. max. file length

SIN JSYS 52

Does string input from a source

ACCEPTS IN 1: TENEX source designator

2: a string pointer to the caller's address space

3: Ø => Ø byte terminates
<Ø => negative byte count
>Ø => a positive byte count or use contents of 4 for terminating byte (whichever happens first)

4: optional right justified byte on which to terminate input

SIN

RETURNS +1: always, updates string pointer in 2 and in 1 (if pertinent) and updates count in 3 (if pertinent)

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual). Updates the file position pointer for subsequent I/O to this JFN. If interrupted by an end-of-file PSI, 1, 2, and 3 will be updated (where pertinent) to reflect the number of bytes transferred before the end-of-file.

SIN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	not open
IOX1:	illegal read
IOX4:	end of file
IOX5:	data error

SOUT JSYS 53

Does string output to a destination

ACCEPTS IN 1: a TENEX destination designator

2: a string pointer from caller's address space

3: Ø => Ø byte terminates
<Ø => negative byte count
>Ø => a positive byte count or use contents of 4 for terminating byte (whichever happens first)

4: optional right justified byte on which to terminate output

SOUT

RETURNS +1: always, updates string pointer in 2 and in 1 (if pertinent) and count in 3 (if pertinent)

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual). Updates the file position pointer for subsequent I/O to this JFN.

SOUT ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX5:	file not open
IOX2:	illegal write
IOX5:	data error
IOX6:	attempt to write beyond abs max file length

RIN JSYS 54

Does random input from a file

ACCEPTS IN 1: JFN

3: the byte number within the file

RIN

RETURNS +1: with byte right justified in 2

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual). Updates the file position pointer for subsequent I/O to this JFN.

RIN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer illegal
DESX5:	file not open
IOX1:	illegal read
IOX3:	illegal to change pointer
IOX4:	end of file
IOX5:	data error

ROUT JSYS 55

Does random output to a file

ACCEPTS IN 1: JFN

2: a right justified byte

3: the byte number within the file

ROUT

RETURNS +1: always

Can cause several pseudo-interrupts or fork termination on certain file conditions (see Section 5 of this manual). Updates the file position pointer for subsequent I/O to this JFN.

ROUT ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer illegal
DESX5:	file not open
IOX2:	illegal write
IOX3:	illegal to change pointer
IOX5:	data error
IOX6:	attempt to write beyond abs max file length

PMAP JSYS 56

JSYS for mapping pages

ACCEPTS IN 1: LH: the source fork/file handle
RH: a page number
OR if 1 contains -1 means remove page (specified in 2) from fork or file (specified in 2)

2: left half: the destination fork/file handle
right half: a page number

3: Access requested as follows
B2 read access allowed
B3 write access allowed
B4 execute access allowed
B8 trap-to-user on any access
B9 copy-on-write

PMAP

RETURNS +1: always

In general PMAP changes the map of the destination fork/file (LH2) so that references to the indicated page go to the source page (RH1). Details vary as described under the following 3 cases.

CASE 1: AC 1 identifies file, AC 2 identifies a fork

This changes the contents of the map of the process such that addresses in the page specified by the right half of 2 will actually reference the indicated page of the file. This is done by setting up a share pointer in the process map to the proper slot in the file's page map. If the file page is non-existent, it will be created and a disk address assigned.

The read, write, and execute permission bits are "and"ed with the permission specified when the file was opened and put into the process map. This "and"ing operation can only reduce access capabilities, it will not generate error conditions. If the file was opened with "access as specified in page table", then the access granted is the "and" of the access requested and that in the page table of the file.

CASE 2: AC 1 identifies a fork, AC 2 identifies a file

This is the inverse operation of case 1. The page (which must be a private page or PMAPX2 error will occur) changes ownership from the fork to the file. The fork will still have the designated page in its address space, but it will have a share pointer to the page in the designated file. The disposal bits are irrelevant. The fork's access to the page will be modified (possibly reduced) in the same manner as described in case 1. If a page already existed at the designated page of the file, the former page will be deleted. (Note this designated page of the file must have been private to the file or PMAPX1 error will occur)

CASE 3: both AC's identify forks

In this case an indirect pointer is set up from the destination page table to the source page table. If the destination page previously existed, it is removed from the map. The access bits for the indirect pointer are set up exactly as specified in 3, but the paging hardware will "and" the access bits at each stage of indirection which may result in a reduction of the final access granted to the destination page.

ILLEGAL CASES

If both AC's designate files or both contain Ø, the PMAP is considered illegal.

PMAP generates an illegal instruction PSI on error conditions listed below, and can potentially generate other PSIs on certain file conditions.

Note on write-copy and trap-to-user bits

a) The monitor will permit a write operation into a page if either write or write-copy access is indicated. A write-copied private page will always have write access, with read and execute present if they are present in the original pointer.

b) The monitor will never copy a private page.

c) GET opens files for read and execute access only. Therefore write access will never be found in a shared (SSAVE'd) page as the result of a GET. Write-copy may be on as determined by the arguments to SSAVE at the time the file was saved.

d) A memory reference to a page having trap-to-user set will:

   1. Clear the trap-to-user bit and continue the reference.

   2. Request a pseudo-interrupt on channel 21.

e) Trap-to-user is noticed by the monitor before other access restrictions. Illegal read and write are next, followed by copy-on-write.

PMAP ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
DESX5:	file not open
DESX7:	file must be on disk
PMAPX1:	attempt to gain illegal access to file
PMAPX2:	attempt to execute PMAP with one of illegal cases
FRKHX1:	illegal fork handle
FRKHX2:	cannot manipulate a superior fork
FRKHX3:	cannot reference multiple forks

RPACS JSYS 57

Reads the accessibility of a page

ACCEPTS IN 1: LH: fork/file designator
RH: a page number in the file

RPACS

RETURNS +1: always with access information in 2

B2 read access allowed
B3 write access allowed
B4 execute access allowed
B5 page exists
B6 indirect pointer (1 or more)
B8 trap-to-user
B9 copy-on-write
B1Ø private

B2Ø read access allowed in first pointer
B21 write access allowed in first pointer
B22 execute access allowed in first pointer
B23 page exists in first pointer
B26 trap-to-user in first pointer
B27 copy-on-write in first pointer

Note that the left-half bits are the result of tracing any indirect pointer chains; whereas, the RH contains information about the first pointer only (the one in the map directly indicated by the argument). The LH and RH information will be different only if an indirect pointer was encountered in the first map. In this case, bit 6 will be set, the LH access will be less than or equal to the RH access, and the trap-to-user and copy-on-write bits will be set in the LH if they were found set at any level. Bits 5 and 1Ø always refer to the last pointer (i.e. first non-indirect pointer) encountered.

Generates illegal instruction PSI on error conditions below.

RPACS ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: file not open
DESX7: file must be on disk
FRKHX1: illegal fork handle
FRKHX3: cannot reference multiple forks

SPACS JSYS 6Ø

Sets the accessibility of a page

ACCEPTS IN 1: left half: fork/file designator
right half: a page number in the file

2: access information

Affects only the map word directly indicated, i.e. no indirect pointers are followed.

B2 permit read access
B3 permit write access
B4 permit execute access
B8 trap to user (PSI channel 21) on any access
B9 copy on write

SPACS

RETURNS +1: always

When modifying a fork page, SPACS will allow no greater access than could be obtained with PMAP. Open file protection is applied to SPACS operations involving file pointers.

SPACS will never allow any bits to be set in a page which does not already exist.

Generates illegal instruction PSI on error conditions below.

SPACS ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
DESX5:	file not open
DESX7:	file must be on disk
SPACX1:	attempt to gain illegal access to a page
FRKHX1:	illegal fork handle
FRKHX2:	cannot manipulate a superior fork
FRKHX3:	cannot reference multiple forks

RMAP JSYS 61

Acquires a handle on a page in a fork

ACCEPTS IN 1: left half: a fork handle
right half: a page number in that fork

RMAP

RETURNS +1: with a handle on the page in 1
BØ-B17, a fork/file designator
B18-B35, a page number

Returns in 2: Access information as follows

B2 read access allowed
B3 write access allowed
B4 execute access allowed
B5 page exists
B8 trap-to-user on any access
B9 copy-on-write

If the argument to RMAP specifies a page in a fork (not a file), and that page is private, then the page will be moved to the job's PMF and will then be shared between the fork and the file. The identifier returned in this case will contain the JFN of the job's PMF and the page number in that file that was assigned.

NOTE: To just find out whether a page exists, use RPACS which will tell you without changing the state of any type of page.

If the specified page is shared with a file but no JFN is assigned to the file (e.g. because it was released with the B5 option of CLZFF), -1 is returned in 1 and Ø in 2.

Generates illegal instruction PSI on error conditions below.

RMAP ERROR MNEMONICS:

FRKHX1: illegal fork handle

SACTF JSYS 62

Sets account number or string to which existing file is being charged.

ACCEPTS IN 1: JFN

2: account number in bits 3-35 if bits Ø-2 = 5 else a string pointer to an account string in the address space of caller. If the normal terminating null byte is not seen, the string is terminated after the 8th full word is processed.

SACTF

RETURNS +1: illegal, error # in 1

+2: successful, updated string pointer in 2

Account must be numeric if LOGIN directory takes numeric accounts only, regardless of directory JFN is in, unless the process has the WHEEL or OPERATOR special capability enabled.

SACTF ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
SACTX1:	file is on non-multiple directory device
SACTX2:	no room in JSB for manipulating string
SACTX3:	alphanumeric accounts not allowed in this directory
SACTX4:	access not allowed

FILE DESCRIPTOR BLOCK FORMAT

The field "CHANGE" lists who can change particular parameters.
w=wheel, op=operator, or=anyone with ownership rights,
wa=anyone who has write access to the file.
if blank, no one can change the field with CHFDB JSYS.

N	NAME	NWORDS	CHANGE	DESCRIPTION
Ø	none	1		the header
1	FDBCTL	1		LH: control bits
			or	BØ, FDBTMP file is temporary
			w+op	B1, FDBPRM file is permanent
				B2, FDBNEX no extension for this FDB yet, file does not really exist
			w+op+or	B3, FDBDEL file is deleted
			w+op	B4, FDBNXF file does not exist (first write not complete)
				B5, FDBLNG long file
				B6, FDBSHT compressed page table
			w+op+or	B17, FDBEPH ephemeral file
				RH: location of file name block
2	FDBEXT			LH: location of extension block
				RH: pointer to other extensions
3	FDBADR	1		the file address & class field
4	FDBPRT	1		file protection word
				LH: 5ØØØØØ
			w+op+or	RH: file protection bits
5	FDBCRE	1	w+op	creation date and time of version 1
6	FDBUSE	1	w+op	LH: last write directory number
				RH: use count (+1 for each indirect pointer and saved environment)
7	FDBVER	1		LH: version number
				RH: pointer to next version
1Ø	FDBACT	1		account information for charging
				+ for location of string block
				- for number
11	FDBBYV	1	or	LH: number of versions to retain in BØ-B5
			or+wa	byte size B6-B11
				RH: number of pages in file
12	FDBSIZ	1	wa	byte count which would address EOF
13	FDBCRV	1	w+op	creation date and time of this version
14	FDBWRT	1	w+op	date and time of last write
15	FDBREF	1	w+op	date and time of last read
16	FDBCNT	1	w+op	LH: count of writes
			w+op	RH: count of reads
17	FDBBCK	5		words for backup system
24	FDBUSW	1	or	user settable word

GTFDB JSYS 63

Gets a file descriptor block

ACCEPTS IN 1: JFN

2: XWD number of words to read, displacement in FDB of first word to read.

Example: XWD 25,0
would read the whole block

3: starting location in address space of caller for reading in descriptor block

GTFDB

RETURNS +1: always

Generates Illegal Instruction PSI on error conditions below.

GTFDB ERROR MNEMONICS:

GFDBX1:	displacement illegal
GFDBX2:	number of words illegal
GFDBX3:	list access not allowed
DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal

CHFDB JSYS 64

Changes selected words in a file descriptor block

ACCEPTS IN 1: LH: displacement into FDB indicating word to be changed
RH: JFN

2: mask indicating bits to be changed

3: changed bits

CHFDB

RETURNS +1: always

Generates Illegal Instruction PSI on error conditions below

CHFDB ERROR MNEMONICS:

CFDBX1:	displacement illegal
CFDBX2:	illegal to change specified bits
CFDBX3:	must be operator or wheel to change specified bits
CFDBX4:	illegal to set specifed bits to value in 3
DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal

DELDF JSYS 67

Reclaims space (expunges) files which have been marked for deletion on DSK: with DELF

ACCEPTS IN 1: directory number

DELDF

RETURNS +1: ALWAYS

Directory number given must be that of connected directory unless process has WHEEL special capability enabled.
If call is in any way illegal, DELDF does nothing.

DUMPI JSYS 65

Reads into core in an unbuffered data mode (mode 17)

ACCEPTS IN 1: JFN

2: pointer to a command list in core

DUMPI

RETURNS +1: unsuccessful, error # in 1
pointer to offending command in 2

+2: successful, pointer in 2 updated to point to last command

COMMAND LIST FORMAT:

Three types of entries may occur in the command list.

a. IOWD n, loc - Causes n words from loc through loc+n-1 to be transmitted. The next command is obtained from the location following the IOWD. The assembler pseudo-op IOWD generates XWD -n, loc-1.

b. XWD Ø, y - Causes the next command to be taken from location y. Referred to as a GOTO word. Up to three consecutive GOTO words are allowed in the command list. After three consecutive GOTO words, an I/O instruction must be written.

c. Ø - Terminates the command list.

DUMPI ERROR MNEMONICS:

DUMPX1:	command error
DUMPX2:	data mode not 17 for this JFN
DUMPX3:	address error(too big or crosses end of memory)
DUMPX4:	access error(cannot read or write data in memory)
DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
DESX5:	not open
IOX1:	illegal read
IOX4:	end of file
IOX5:	data error

DUMPO JSYS 66

Writes from core using an unbuffered data mode (mode 17).

ACCEPTS IN 1: JFN

2: pointer to a command list in core
(see DUMPI)

DUMPO

RETURNS +1: unsuccessful, error # in 1
pointer to offending command in 2

+2: successful, pointer in 2 updated to point to last command

DUMPO ERROR MNEMONICS:

DUMPX1:	command error
DUMPX2:	data mode not 17 for this JFN
DUMPX3:	address error(too big or crosses end of memory)
DUMPX4:	access error(cannot read or write data in memory)
DESX1:	illegal value for designator
DESX2:	terminal not available to this job
DESX3:	no name for this JFN
DESX4:	terminal designator or string pointer not legal
DESX5:	not open
IOX2:	illegal write
IOX5:	data error

MTOPR JSYS 77

Performs device dependent control functions.

ACCEPTS IN 1: JFN

2: operation wanted (see table below)

MTOPR

RETURNS +1: successful

Generates Illegal Instruction PSI on error conditions below

For magnetic tape (MTA:), operations are:

1	rewind
3	write eof
6	forward space one record
7	backward space one record
10	space to end of tape
11	rewind and unload
13	write 3 inches of blank tape
16	forward space one file
17	backward space one file

For the ARPA network (NET:), operations are:

20	Accept connection. Causes RFC to be sent.	
21	Dump buffer. Sends all currently buffered bytes.	
22	Send INS/INR	
23	not used	
24	Assign interrupt	
	AC3 0-5	INS/INR PSI channel (≥ 36 means don't interrupt)
	6-11	not used
	12-17	State change PSI channel
	18-35	Reserved for expansion

For DECTAPE (DTA:), operations are:

1	Rewind
11	Rewind and flap tape
30	Use block # in AC3 for next DUMPI/O.

All other operations are NOPs except that all MTOPRs clear error and end-of-file flags.

MTOPR ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: not open

ATPTY JSYS 274

Attaches a pair of network connections to a network terminal.

ACCEPTS in 1: JFN of opened but unused receive connection
2: JFN of opened but unused send connection

ATPTY

RETURNS +1: unsuccessful, error # in 1
+2: successful, terminal designator in 1

ATPTY ERROR MNEMONICS:

ATPX1: invalid receive JFN
ATPX2: JFN in 1 is not receive connection
ATPX3: receive JFN not opened
ATPX4: receive JFN not network connection
ATPX5: receive JFN has been used
ATPX6: receive connection refused
ATPX7: invalid send JFN
ATPX8: JFN in 2 is not send connection
ATPX9: send JFN not opened
ATPX1Ø: send JFN not network connection
ATPX11: send JFN has been used
ATPX12: send connection refused
ATPX13: NO PTY's available

GPJFN JSYS 206

Get primary JFN's

ACCEPTS IN 1: fork handle

GPJFN

RETURNS +1: always
2: LH: primary input JFN
RH: primary output JFN

Gives illegal instruction PSI on the following errors:

FRKHX1: illegal fork handle
FRKHX2: cannot reference superior fork
FRKHX3: cannot reference multiple forks

Primary I/O - If the primary input designator is not exactly the same as the primary output designator (see SPJFN, GPJFN), each character of input will be "echoed" on the output.

SPJFN JSYS 207

Set primary JFN's

ACCEPTS IN 1: fork handle

2: LH: primary input JFN
RH: primary output JFN

SPJFN

RETURNS +1: always

JFN cannot be 100 or 101: either of these will cause the specified fork to get an error on any primary I/O operations.

Gives illegal instruction PSI on the following errors:

FRKHX1: illegal fork handle
FRKHX2: cannot reference superior fork
FRKHX3: cannot reference multiple forks

for expansion of the network section. pages 72,73,74,75,76,77,78,79,8Ø.

ESTABLISHING AND BREAKING NETWORK CONNECTIONS

User programs that use the ARPANET do so through the TENEX file system. Establishing a network connection from TENEX involves two steps:

obtaining a JFN for a "file" corresponding to the desired connection (via the GTJFN JSYS) and;

transmitting a "request for connection" (RFC) to the appropriate remote Host (via the OPENF and/or MTOPR JSYS'S)

A network connection has the following characteristics:

Two ends:

Each end of a connection is called a socket. Socket names are 32 bit numbers. The end of the connection in the local Host computer is called the local socket and the end in the remote Host, the foreign socket. The remote computer is called the foreign Host. Only one connection with the same pairs of ends may be in existence at a given time.

A byte size:

Data transmission over the network occurs in units of bytes. The byte size for a connection is fixed when the connection is established. For TENEX, lack of agreement in byte size with the foreign host will cause the connection attempt to fail.

A direction:

Information may flow in only one direction over a connection.

In TENEX, file names corresponding to network connections are of the form:

NET:[LS[#]].[FH-FS][;T]

where [ ] is used to indicate optional presence, LS specifies a local socket, FH a foreign host and FS a foreign socket.

Device field:

NET: indicates that the "file" corresponds to a network connection.

Name field:

LS identifies a local socket. It should be an octal number less than 400 (8 bits) unless "#" is present in which case it should be less than 40000000000 (32 bits). When LS is omitted, its default value is taken to be JFN*10 (octal), where JFN is the job file number returned by GTJFN. The actual (absolute) local socket corresponding to LS is determined as follows:

If "#" is used, the local socket is LS. Use of "#" requires wheel or operator capability;

If ";T" is used, the local socket is said to be "job relative" and is guaranteed to be unique to the user's job. In the current implementation, the local socket number is:

(303240 + JOB) LS

high 17 bits low 15 bits

where JOB is the user's TENEX job number.

If ";T" is not used, the local socket is said to be "directory" or "user" relative and is guaranteed to be unique to the connected directory. In the current implementation, the local socket number is:

DIR LS

high 17 bits low 15 bits

where DIR is the number of the connected directory.

Extension Field:

FH and FS identify the foreign Host and socket. FH should be either the official ARPANET name (or nickname) for the foreign Host or the network address (in octal) of the foreign Host. A table of Host names and addresses appears in the manual "ARPANET TENEX". FS specifies the (absolute) foreign socket and should be an octal number less than 40000000000 (32 bits). If the extension is null, the subsequent OPENF causes the connection to be placed in the so-called "listening" state.

Examples:

NET:1#.	specifies the local logger socket.
NET:1Ø.UTAH-1Ø-1	specifies a directory relative connection to the UTAH logger socket.
NET:11.1Ø6-4ØØ346;T	specifies a job relative connection to socket 4ØØ346 at host number 1Ø6 (octal), which is MIT-DMCG.

The OPENF JSYS is used to open network connections.

Connection byte sizes may be 8, 32 or 36. Connections opened for reading are receive connections and those opened for writing are send connections. If non-zero, bits 12-17 of AC2 specify that the buffer for the connection is to be smaller than the system default and of the specified size in words.

There are four legal data modes for network "files" which independently specify strategies for connection establishment and data transmission. The modes are:

Ø Normal mode. Wait until a matching RFC or CLS is received from the remote Host before returning from OPENF to the user program. For data transmission, send bytes as soon as possible.

5 Buffered send mode. Wait (as above) before returning to the user program. For transmission, data is to be held in local system buffers until an amount appropriate for efficient network utilization accumulates or until the user program explicitly requests transmission (via the MTOPR JSYS described below).

6 Immediate return mode. Return from the OPENF immediately without waiting for matching RFC or CLS from the foreign host. The connection must be "open" before data transfer can be successfully attempted. Methods for determining the "state" of a connection are described below.

7 Buffered send and immediate return. This mode combines modes 5 and 6.

For a network connection OPENF may fail if:

The foreign Host is dead (OPNX2Ø);

The connection already exists (OPNX9);

The connection attempt is rejected by the foreign host (OPNX21);

The local system's network connection table has insufficient space for a new connection (OPNX1Ø).

The specified byte size does not match that of the RFC specification received (for receive connections only) (OPNX22)

A connection whose corresponding file name has a null extension is placed in a listening state by OPENF. In the listening state it is receptive to RFC's from any socket at any site. Arrival of such an RFC results in a change of connection state (which may cause an interrupt; see MTOPR), at which point the user program may choose to accept the connection attempt by causing a matching RFC to be sent (via a data transfer operation, e.g. BIN, BOUT, or via the MTOPR JSYS) or it may choose to reject it by issuing a CLS (via CLOSF). To make this decision, GDSTS may be used to determine the remote host and socket numbers.

Network connections may be closed by any of the JSYS'S that close files (CLOSF, RESET, etc.). An option available when closing a connection via CLOSF is whether to wait until the connection is "completely" closed. When bit 1 of AC1 is set, CLOSF does not return to the user program until a matching CLS has arrived from the foreign Host. This option is useful when it is desired to immediately reopen the connection; if the connection is not completely closed when the next attempt to open it is made, the attempt will fail.

DATA TRANSFER OVER NETWORK CONNECTIONS

The standard TENEX file data transfer JSYS's, BOUT, SOUT, BIN, and SIN, are used to send and receive data over network connections.

BOUT and SOUT cause data to be moved from the user's address space to TENEX system network buffers for transmission. For connections opened in unbuffered send mode (see above), the data is sent out into the network as soon as possible (consistent with Host-Host flow control restrictions). Use of unbuffered send connections can result in a large number of small network messages. More efficient use of the network can be made by using connections opened in buffered send mode for which TENEX collects the data into larger network messages for transmission. TENEX does this by keeping the data within system buffers until "enough" accumulates to allow for efficient network utilization before sending it to the remote Host. The user program can force TENEX to transmit the accumulated data using the MTOPR JSYS.

If the system buffers for a connection are full (and HOST-HOST flow control restrictions prevent TENEX from transmitting the data), BOUT and SOUT will "hang". This can happen if the remote process reading from the connection removes data more slowly than it is being sent. Similarly, if the system buffers for a connection are empty, BIN and SIN will "hang". To avoid hanging on input, the SIBE JSYS can be used to test the condition of the buffers before attempting the data transfer.

It sometimes happens that connections that have been established are unexpectedly broken; e.g. the remote Host crashes or the remote process closes the connection without warning. An attempt to send over a connection that has been closed results in an IO Data Error PSI. Attempting to read from a connection that has been closed delivers null bytes and results in an End of File PSI.

OBTAINING STATUS INFORMATION

Information about a network connection may be obtained via the GDSTS JSYS. For a network file the following values are returned in the designated accumulator.

2: BØ-B3: Contains connection state

3: Contains the foreign host number

4: Contains the foreign socket number

The states a connection may be in are:

Ø	DEAD	Not possible if an OPENF has been done for the JFN.
1	CLZD	Cannot be returned by GDSTS.
2	PNDG	Cannot be returned by GDSTS.
3	LSNG	The name corresponding to the JFN had a null extension. An OPENF has been done and no attempt has been made to connect to the local socket.
4	RFCR	The name corresponding to the JFN had a null extension. An OPENF has been done and a request for connection has been received for the local socket from some foreign host.
5	CLW2	Cannot be returned by GDSTS.
6	RFCS	An OPENF has been done specifying a foreign host-socket and a response has not yet arrived from the foreign host.
7	OPND	The connection is fully open and data transmission may occur.
1Ø	CLSW	For a connection with a JFN, this state means that a time out has occurred for a connection which was in the RFCS state and for which the foreign Host has not yet responded with the proper close commands.
11	DATW	Cannot be returned by GDSTS
12	RFN1	Cannot be returned by GDSTS

13 CLZW The foreign Host has requested that the connection be closed.

14 RFN2 This is a transitory state which may occur before CLZW, if there was an outstanding RFNM when the foreign host requested the connection to be closed.

16 FREE Cannot be returned by GDSTS.

The Pseudo-Interrupt system can be set up to generate an interrupt when the state of a specified connection changes or an INS or INR is received. To do that the MTOPR JSYS (described below) is used to specify a connection and PSI channels.

INR/INS Interrupts and the MTOPR JSYS

The ARPANET Host-Host protocol includes a mechanism for transmitting "interrupt" signals over connections. It provides for "interrupt by sender" (INS) and "interrupt by receiver" (INR). A connection interrupt signal bypasses the Host-Host flow control restrictions for data on the connection and therefore will get through to the remote host even when the connection data buffers are full. The INS/INR mechanism is used in some server TELNET implementations for "attention" signalling. The Host-Host protocol interrupt mechanism is accessible to TENEX user programs via the MTOPR JSYS.

The MTOPR JSYS is used to perform a variety of device dependent control functions. For the "network device" (NET:) MTOPR may be used to force transmission (for buffered send connections), to send an RFC, to send INS/INR signals and to set up the PSI system to handle change of state and INR/INS interrupts.

MTOPR functions for the network files are:

2Ø Accept connections. Causes an RFC to be sent (only valid if connection in RFCR state)

21 Dump buffer. Sends all currently buffered bytes

22 Send INS/INR

24 Assign Interrupt

AC3: BØ-B5 INS/INR PSI channel
(> 36 means don't interrupt)
Note that the indicated channel and the rest of the PSI system must also be initialized and enabled and activated.
B6-B11 not used
B12-B17 State change PSI channel
B18-B35 Reserved for expansion

MISCELLANEOUS

TENEX does not make an entry in its network connection tables for a connection until OPENF is performed. The least significant bit of both LS and FS in a network file name is ignored until the OPENF, at which time the correct low order bit, as required by the Host-Host protocol, is inserted. For "send" sockets the least significant bit is set to 1, for "receive" sockets, to Ø.

CVSKT JSYS can be used to determine the absolute local socket number corresponding to a JFN.

CVSKT JSYS 275

Converts a local network socket number to absolute form.

ACCEPTS IN 1: JFN

CVSKT

RETURNS +1: unsuccessful, error # in 1
+2: successful, absolute local socket number in 2.

CVSKT ERROR MNEMONICS:

CVSKX1:	Bad JFN
CVSKX2:	name field associated with JFN does not decode into a reasonable local socket specification

A list of host names and numbers will be found in the new manual "ARPANET TENEX".

CVHST 276

ConVerts a HoST number to primary name.

ACCEPTS IN 1: TENEX destination designator where ASCIZ string is to be written

2: host number

CVHST

RETURNS +1: failure (no string for that number)

+2: successful, with host name written.

LHOSTN is a single entry system table which may be read with the SYSGT JSYS. It contains the host number corresponding to the local TENEX.

FLHST JSYS 277

FLush HoST

ACCEPTS IN 1: host number

FLHST

RETURNS +1: always

Resets the NCP tables which contain status information about the indicated host, and sends host-to-host protocol RST (reset) command to that host.

Has no effect unless the user is an ENABLE'ed WHEEL or OPERATOR.

Section 2 Index

JSYS's FOR OBTAINING INFORMATION

INTRODUCTION

These JSYS's 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 (intervals, accumulated times, etc.). These are all integer numbers in units of 1/N seconds, where N is provided by the TIME JSYS. The system interval clock is in milliseconds, and the current value of N is therefore 1000.

GJINF JSYS 13

Gets job specific information

GJINF

RETURNS +1: always with accumulators listed below containing:
1 directory # under which job is logged in
2 directory # to which job is connected
3 tss job number
4 terminal number attached to job or -1 if job is detached

TIME JSYS 14

Gives time since system was restarted last.

TIME

RETURNS +1: with time right justified in 1,
divisor to convert to seconds in 2

This is a monotonically increasing number (when the system is running) independent of resets of the time and date and should be used for timing execution of programs, and for getting actual time and date.

RUNTM JSYS 15

Gives runtime of one process or the whole job

ACCEPTS IN 1: a fork handle (-5 means whole job)

RUNTM

RETURNS +1: with runtime right justified in 1,
divisor to convert to seconds in 2,
console time in 3 (apply divisor in 2 to get seconds)

Generates illegal instruction pseudo-interrupt on error conditions below.

RUNTM ERROR MNEMONICS:

FRKHX1: illegal fork handle
RUNTX1: fork handles -3,-4 are illegal

JOBTM JSYS 316

Gives runtime in milliseconds at the current job time.

JOBTM

RETURNS +1: always, with the runtime in milliseconds of the current job in 1.

SWTCH JSYS 32Ø

Gives the 36 data switch settings

SWTCH

RETURNS +1: always, with the 36 data switch settings in 1

GTAD JSYS 227

Gets current time and date in system internal format (see introduction).

GTAD

RETURNS +1: with date in LH 1,
time in seconds since midnite in RH 1.

-1 is returned in 1 if the system does not have the time and date.

STAD JSYS 226

Sets system time and date

ACCEPTS IN 1: date and time

STAD

RETURNS +1: unsuccessful, error code in 1

+2: successful

Requires WHEEL or OPERATOR special capability if system already has date and time.

STAD ERROR MNEMONICS:

STADX1: date and time already in system and caller does not have WHEEL or OPERATOR capability enabled

STADX2: impossible values for date and time

The ensuing two JSYS's, ERSTR and GETER, translate error numbers to error strings and retrieve error numbers. A file called ERROR.MNEMONICS in the SYSTEM directory in TENEX contains the error message strings associated with system error numbers. The error message strings are ASCII strings terminated by @, and they have a simple structure which permits them to type out the parameters of the error. To facilitate this, five cells are set aside in the PSB for storing error parameters. These parameters include such things as the location of an offending OP-CODE, an illegal character presented in a file name, etc. The error message string has a simple command language which enables one to specify various modes of typeout of particular error parameters.

A parameter typeout command is denoted by a "%" character followed by one or more commands terminated by a "%" character. The character "%" may not be used as a character in the text of error message strings. The character "\" is also illegal since it is used as the terminator to ASCII (see example).

The following commands are defined after %:

The n stands for the particular word in the PSB to be used as the parameter.

- nA types parameter as ASCII (null terminates)
- nO types parameter in octal
- nD types parameter in decimal
- nH types parameter in octal half words separated by comma
- nF types parameter in floating point free format output
- mE is a recursive pointer to another error message (m is the error number)
- L means type out enough information to locate this process in the fork structure. A typeout of the form 1.2.4.2 will result which is a list of the relative fork handles by which one could reference this fork starting at the top level.
- nN means interpret contents of word n in PSB as a JFN which means to fill in the file name associated with this JFN
- @nZ (where Z stands for any of the commands above) The @ means use the location in this fork pointed to by the cell in the PSB

An example of an error string would be the illegal instruction error string:

ASCII\ Attempt to execute the illegal instruction %ØH% at location %10% of fork %20%@\

ERSTR JSYS 11

JSYS to translate an error # to a string

ACCEPTS IN 1: TENEX destination designator

2: LH: a fork designator
RH: an error number (-1 means the most recent error)

3: LH: a negative count of maximum number of bytes in string to be transferred, or Ø for no limit
RH:
B18 off means expand parameter typeout commands, on means do not expand
B19 off means use 5 words in PSB of designated fork, on means use accumulators 4-1Ø

4-1Ø: used optionally, see 3

ERSTR

RETURNS +1: undefined error number

+2: string size out of bounds or illegal TENEX destination designator or illegal fork handle

+3: successful

NOTE: The ESOUT JSYS (Section 2) should be used to report errors in consistent fashion.

GETER JSYS 12

JSYS to get the most recent error condition encountered in a fork

ACCEPTS IN 1: fork designator

GETER

RETURNS +1: always, with most recent error condition in RH of 2 (always) and fork designator in LH of 2, and with 5 parameters from PSB in 4-1Ø

The most recent error condition is always saved in PSB.

NOTE: GETER differs from most JSYS's in that it modifies accumulators higher than 4.

SETNM JSYS 21Ø

Declares name of subsystem being used by job, i.e., sets name.

ACCEPTS IN 1: SIXBIT name used to identify program or subsystem

SETNM

RETURNS +1: always

Subsystem usage statistics are accumulated in system tables SNAMES, STIMES, and SPFLTS (defined elsewhere in this section). To make this possible, SETNM must be executed by each job whenever the subsystem is changed. In all usual cases, the EXEC handles this. The argument to SETNM should be: for subsystems (programs from directory<SUBSYS> or <SYSTEM>), the file name, truncated to six characters and converted to SIXBIT; for all other programs, "(PRIV)".

NOTE: Each program named by a SETNM takes a slot in the system statistics tables. When all slots are used, the names will be set to .OTHER regardless of the contents of AC 1.

Please do not execute SETNM unless you need statistics for the program.

GETNM JSYS 177

Gets name of subsystem being used by job. (Previously declared by SETNM)

GETNM

RETURNS +1: always, with
SIXBIT name of program in 1

GTRPI JSYS 172

Get trap information for a specific fork.

ACCEPTS IN 1: a fork handle

GTRPI

RETURNS +1: always, with:

1. number of pager traps for designated fork
2. number of page faults for designated fork
3. time spent in page routines by designated fork

GTRPI ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

GTDAL JSYS 3Ø5

Gets the disk allocation of a specified directory.

ACCEPTS in 1: desired directory (nonzero value not implemented yet)
Ø=> currently connected directory.

GTDAL

RETURNS +1: always, with allocated number of pages for the specified directory in 1, and number of pages currently used in that directory in 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 1-word entries. Entry -1 in each table is the negative of the number of data entries in the table; the data entries are identified by an index which runs up from Ø. Two JSYS's exist for accessing tables: the first, SYSGT, takes a table name and returns the table length, its entry Ø, and a number identifying the table for use with the other JSYS, GETAB, to obtain additional entries from the table.

Where numeric table indices are given, they are octal:

name	index	contents
JOBTTY	TSS JOB #	LH: controlling terminal line #, -1 if none (i.e., job is detached) RH: top fork in job
JOBRT	TSS JOB #	CPU time used by job, negative if no such job
TICKPS	one-word table giving divisor to convert JOBRT entries to seconds	
JOBDIR	TSS JOB #	RH: logged in directory number, Ø if not logged in LH: connected directory number
TTYJOB	line number	LH: positive TSS job # for which this is controlling terminal, or -1 for unassigned line, or -2 for line in process of being assigned, or 4ØØØØØ+TSS job # to which this line is assigned. RH: -1 if no fork is waiting for input from this terminal, other than -1 if some fork is waiting for input.
NCPGS	one-word table containing number of pages of real (physical) user core available in system	

Tables DEVNAM, DEVCHR, and DEVUNT are parallel; their indices are significant only in that the same entry in each table pertains to the same device.

DEVNAM SIXBIT device name, e.g., "DTA1"

DEVCHR device characteristics word, as documented under DEVCHR JSYS in section 4, except that B5 is not meaningful.

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

DSKERR information on disk errors:

Ø	number of recoverable disk errors
1,2,3	command words for last recoverable disk error
4	error bits for last recoverable disk error
5	number of irrecoverable disk errors
6-11	3 command words and error bits for last irrecoverable disk error

DRMERR drum error information

Ø	number of recoverable errors
1,2,3	2 command words and error bits for last recoverable error

SYSVER contains an ASCII string identifying the system name and version.
e.g., BBN-TENEX 1.31.23

VERNUM One-word table containing the system version as a number, e.g., the decimal number 13123.

PGSTAT Pager trap information for this process

Ø	count of pager traps
1	count of page faults
2	time spent in trap routines

SYSTAT monitor statistics

Ø	time with no runnable jobs (in ticks, see TICKPS)
1	waiting time with 1 or more runnable jobs (waiting for page swapping)
2	time spent in core management
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
1Ø	number of terminal wakeups
11	number of terminal interrupts
12	time integral of number of jobs in balance set
13	time integral of runnable jobs
14	1 min. load average (floating point format)
15	5 min. load average (floating point format)
16	15 min. load average (floating point format)
17	sum of process disc wait
2Ø	sum of process drum wait
21	total number terminal input characters
22	total number terminal output characters
23	total number of garbage collections
24	integral postpurge time
25	count of forced balance set removals
26	sum process wait time
27	time of pending system shut down (TENEX internal time format)
3Ø	time system scheduled back up (TENEX internal time format)

Note: This table is subject to change (usually additions) as measuring routines are added to the system. Consult monitor symbolic listing SCHED.MAC for most recent description of this table.

QTIMES Ø-4 Accumulated runtime of jobs on the 5 scheduler queues

JOBNAM TSS JOB # LH: unspecified
RH: index into the tables SNAMES, STIMES and SPFLTS for the subsystem being used by this job (determined by the last SETNM JSYS executed by job)

The following 5 tables are parallel: corresponding entries in each table contain information about the same subsystem. The subsystem being run by a specific job may be determined from SNAMES, using an index obtained from table JOBNAM (above).

SNAMES SIXBIT name of subsystem, or Ø if this entry unused in this and the following 2 tables.

STIMES Total runtime of subsystem

SPFLTS Total number of page faults of subsystem.

SWAKES Totals relating to terminal usage, changed often.
SBLKTM See code.

ENTFLG One-word table: non zero if users may log in. If zero, TENEX is not available.

DBUGSW has DBUGSW as its first entry and DCHKSW as its second entry.

SYMTAB points to the table of SIXBIT GETAB names

LOGDES is two words long, the first being LOGDES, the TTY for login/logout messages and the second being JBØTT, the designator of the terminal for error comments from SYSJOB, BUGCHK'S, and the NCP.

The following words and tables relate to the ARPA Network interface, and will not be recognized if the system does not have an IMP connection.

NETRDY Network operational status, 2 words
Ø = -1 if IMP operating normally
1 = Ø if network turned off, non-Ø if on

IMPHRT Table of bits, one for each possible host. Bit N is 1 if on NCP control message has been received for host N. The bit for host N is in bit N mod 36 of word N/36.

HSTNAM gives names of hosts known to NCP, in ASCIZ format.

HOSTN gives information for each host name known to NCP.
e.g. Bit Ø=> server host
1=> user host
3=> nickname
6 - 8=> host type
9 -12=> host name
RH => index of name in HSTNAM

The following parallel tables are indexed by an internal connection number.

NETLSK,NETFSK Local and foreign socket numbers respectively for this connection

NETAWD BØ-8 Link, B9-17 foreign host, B18-26 Internal uses, B27-35 index into IMPLT1 thru IMPLT4.

NETBAL Bit allocation

NETSTS LH Control and Status Bits,
RH Control and Status Bits

NETBUF Line number for terminal connections, otherwise undefined

NETBTC Number of bits sent or received since socket was created.

The following 4 tables are parallel, referenced by another internal index. Their contents are subject to change. For current descriptions of these tables, see the monitor program IMPDV.MAC.

IMPLT1 Internal uses

IMPLT2 Internal uses

IMPLT3 Internal uses

IMPLT4 BØ-19 Internal uses, B2Ø-35 message allocation.

LHOSTN Has the local host number as its first entry, and -n,,m as its second, where n= the number of network PTYs and m= the number of the first PTY.

SYSGT JSYS 16

Gets table number, length, and entry Ø for a named system table, i.e., system get table.

ACCEPTS IN 1: SIXBIT table name

SYSGT

RETURNS +1: ALWAYS
1: entry Ø of table
2: LH: negative of number of entries in table
RH: table number for use with GETAB
or Ø in 2 if table not found

GETAB JSYS 1Ø

Gets a specified word from a (numerically) specified table.

Table numbers should always be obtained at run time with SYSGT; they should not be coded into programs because they will change.

ACCEPTS IN 1: LH: index into table
RH: table number

GETAB

RETURNS +1: unsuccessful, error code in 1

+2: success, 36 bit number from table in 1

GETAB ERROR MNEMONICS:

GTABX1 illegal table number
GTABX2 illegal index
(less than -1 or greater than length of table)

Section 3 Index

SPECIAL DEVICES

INTRODUCTION

This section deals with the various JSYS's for talking to special devices in the system. These devices include Magtape, DECtape, paper tape reader, paper tape punch, line printer, etc. Also included in this section is a discussion of the JSYS's which are actually file system JSYS's but are implemented only for terminals.

Many JSYS's in this section take a TENEX device designator as an argument. This can be:

LH: 600000+device type number
RH: unit number (-1 for non-unit device)

or:

LH: 0
RH: 400000+a terminal number
or 777777 for controlling terminal.

The various devices are:

name	description	type	units?	
DSK:	disc	0		
DRM:	drum	1		can not be explicitly referenced
MTAn:	magnetic tape	2	yes	
DTAn:	DECtape	3	yes	
PTR:	paper tape reader	4		
PTP:	paper tape punch	5		
DSP:	display	6		
LPT:	line printer	7		
CDR:	card reader	10		NOT IMPLEMENTED
CDP:	card punch	11		NOT IMPLEMENTED
TTY:	terminal	12	yes	
TTP:	terminal punch	13		NOT IMPLEMENTED
TTR:	terminal reader	14		NOT IMPLEMENTED
NIL:	nothing device	15		an infinite sink for unwanted output; an infinite source of zeros for input
NET:	ARPA Network	16		
PLT:	Calcomp Plotter	17		

Notes on Some Devices

For some devices, "device dependent status bits" are described. These bits are manipulated with the JSYS'S GDSTS and SDSTS, described in this section.

Line Printer

LPT:

The line printer accepts 128 character codes (Ø-177 octal) each of which normally causes a character to be printed on the line printer as shown in figure 4-1. Exceptions to this are the format control characters as described below. All vertical format controls (except for form feed) are simulated by multiple line feeds for both compatibility with other printers and the increased reliability of not relying on the printer format control tape!

Line printer spooling is in effect in TENEX. Normally output directed to LPT: will be saved on a file and queued for printing. A LPT scheduler selects files for printing based on their length and time-of-write. Very long files (greater than 12Ø,ØØØ characters) will not be printed during defined prime time except as specifically directed by the operators. LP1: opened by an enabled WHEEL or OPERATOR will go direct to the line printer.

CODE	NAME	ACTION
Ø	null	normally prints nothing.
11	tab	skips to the next of every 8th column on the same line.
12	line feed	skips to next line same column (last six lines of each page are skipped)
13	vertical tab	skips to the line at the next third of a page same column.
14	form feed	skips to the top of the next page, same column
15	carriage return	returns to column Ø of the current line, no paper advance.
2Ø	half page	skips to the next half page, same column.
21	alternate lines	skips to the next even line, same column.
22	three lines	skips to the next of every third line, same column.
23	next line	skips to the next line without skipping the last six lines on a page, same column.
24	sixth page	skip to the next sixth of a page, same column
37	end of line	same as carriage return-- line feed.
177	escape	causes any special action of the next character to be ignored. e.g. 177,Ø will print code Ø (unslashed Ø) 177, 177 will print code 177 (hat)

Code	Char	Code	Char	Code	Char	Code	Char
Ø	unslashed Ø	4Ø	space	1ØØ	@	14Ø	`
1	alpha	41	!	1Ø1	A	141	a
2	beta	42	"	1Ø2	B	142	b
3	gamma	43	#	1Ø3	C	143	c
4	delta	44	$	1Ø4	D	144	d
5	epsilon	45	%	1Ø5	E	145	e
6	zeta	46	&	1Ø6	F	146	f
7	theta	47	'	1Ø7	G	147	g
1Ø	lambda	5Ø	(	11Ø	H	15Ø	h
11	mu	51	)	111	I	151	i
12	xi	52	*	112	J	152	j
13	pi	53	+	113	K	153	k
14	rho	54	,	114	L	154	l
15	sigma	55	-	115	M	155	m
16	tau	56	.	116	N	156	n
17	phi	57	/	117	O	157	o
2Ø	psi	6Ø	Ø	12Ø	P	16Ø	p
21	omega	61	1	121	Q	161	q
22	delta	62	2	122	R	162	r
23	gamma	63	3	123	S	163	s
24	pi	64	4	124	T	164	t
25	phi	65	5	125	U	165	u
26	sigma	66	6	126	V	166	v
27	omega	67	7	127	W	167	w
3Ø	dagger	7Ø	8	13Ø	X	17Ø	x
31	tilde	71	9	131	Y	171	y
32	del	72	:	132	Z	172	z
33	arrow	73	;	133	[	173	{
34	infinity	74	<	134	\	174	|
35	integral	75	=	135	]	175	}
36	+-	76	>	136	↑	176	~
37	underline	77	?	137	←	177	hat

Figure 4-1

This is compatible with the DEC PDP-1Ø modified ASCII code with the characters from Ø - 37 as chosen by BBN for BBN'S Anelex printer. Note: 1968 ASCII and many new terminals and printers will have different characters for codes 136 (carat), 137 (underscore) and 176 (tilde).

Magnetic Tape

MTAn:

Magnetic tape device dependent status bits:

B18	illegal write
B19	device error (hung or data late)
B20	data error
B21	suppress automatic error correction
B22	device EOF mark
B23	incorrect record length (not same # words as specified by read op or not whole # words)
B24	beginning of tape
B25	end of tape
B26	parity: 1 = even
B27-28	density: 00=800, 01=200, 10=556, 11=800
B29-31	character counter if B23=1
B32-35	1

Magtape I/O is currently available in TENEX only via the DUMPI/O JSYS's described in section 2. Magtape control functions are performed via the MTOPR JSYS also in section 2.

Paper Tape Reader and Punch

PTR: PTP:

Both reader and punch operate in 4 formats. These are named ASCII, IMAGE, BINARY, and IMAGE BINARY. If the reader or punch is operated in mode Ø, the format is assumed IMAGE if the byte size is 8, BINARY if the byte size is 36, and ASCII otherwise. Any of the 4 formats may be forced by opening in mode 1 for ASCII, mode 1Ø for IMAGE, mode 13 for IMAGE BINARY and mode 14 for BINARY.

The data modes affect operation as follows.

A (ASCII, mode 1) - In ASCII format, nulls are ignored on output; nulls, rubouts, and tapefeed are ignored on input. Each formfeed (character code 14) is followed by 8 lines of tapefeed; each horizontal or vertical tab (character codes 11 and 13) is followed by a rubout. Each character is punched with an odd parity bit in channel 8 and the 7 data bits in channels 7 to 1. The parity is ignored on input.

I (Image, mode 1Ø) - Eight-bit characters are punched exactly as they appear in the buffer with no additional processing.

IB (Image Binary, mode 13) - Binary words are split into six 6-bit bytes and punched with the eighth hole punched in each line. There is no format control or checksumming performed by the I/O routine. Data punched in this mode is read back by the paper-tape reader in the IB mode. Lines of tape without the 8th hole punched are ignored. The 7th hole is ignored.

B (Binary, mode 14) - Each sequence of 4Ø(8) words is punched as one checksummed binary block. Each block is preceded by a header word which contains a count of the data words in the block in the right half, and a 12-bit folded checksum in the left half. The checksum is the 12-bit 1's complement sum of the 3 12-bit bytes of the 2's complement sum of the data words in the block. In BINARY format output, each block except the last contains 32 data words and is followed by 8 lines of tapefeed. Several blank lines are punched after each bufferful for visual clarity.

When the PTP is opened, 128 lines of tapefeed are punched preceding the first line(s) of data. When the PTP is closed, 256 lines of tapefeed are punched following the last line of data.

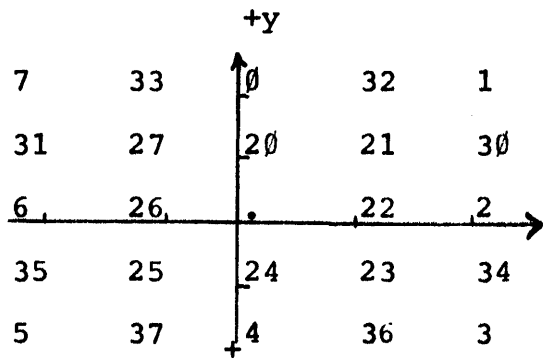
CALCOMP Model 665 Plotter

PLT:

12" wide paper, 2 step sizes

Open PLT: causes about 6" of paper to be slewed. The system routines maintain a "fence" that prevents a user from going off the edge of the paper. The pen is assumed to be centered in the Y direction when PLT: is opened, and users should leave the pen in a similar place when PLT: is closed.

The plotter accepts 5-bit bytes to direct its motion as shown below:



Moving in the +X direction slews paper on to the floor. The +Y direction is arranged so that the coordinate system is right handed.

Standard software (line-drawing, characters, etc.) is available in the FORTRAN library. See other documents for description.

GENERAL

ASND JSYS 7Ø

Assigns a device to caller

ACCEPTS IN 1: a TENEX device designator

ASND

RETURNS +1: unsuccessful, error # in 1

+2: successful (or already assigned to caller).

ASND ERROR MNEMONICS:

DEVX1:	illegal TENEX device designator
DEVX2:	device already assigned to another job
ASNDX1:	device not assignable (eg DSK:)
ASNDX2:	illegal to assign this device
ASNDX3:	no such device

RELD JSYS 71

Releases an assigned device

ACCEPTS IN 1: a TENEX device designator
or -1 to release all devices assigned to this job

RELD

RETURNS +1: unsuccessful, error # in 1

+2: successful

RELD ERROR MNEMONICS:

DEVX1:	illegal TENEX device designator
DEVX2:	specified device assigned to another job

CSYNO JSYS 72

Creates a new name synonym by which the known device can be referenced. The synonym is private to this job and synonyms are searched first. Old name and new name may be used interchangeably.

NOT IMPLEMENTED YET

ACCEPTS IN 1: a string pointer to the known device name or a device designator

2: a string pointer to the new device name

CSYNO

RETURNS +1: unsuccessful, error # in 1

+2: successful, updated string pointers in 1,2

Note that devices can only be renamed to devices. E.g., one cannot rename a device to a disc file!

CSYNO ERROR MNEMONICS:

DESX1:	illegal TENEX source/destination designator
ASNDX3:	no such device
CSYNX1:	synonym already in use

GDSTS JSYS 145

Get device status

ACCEPTS IN 1: JFN

GDSTS

RETURNS +1: always, with device dependent status bits in 2;
AC3: Device dependent. For MTA:, word count of last transfer completed, negative if last attempted transfer unsuccessful.

Currently a NOP for devices without device dependent status bits

Gives illegal instruction PSI on the following errors:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: not open

SDSTS JSYS 146

Set device Status

ACCEPTS IN 1: JFN

2: new status bits

3: Device dependent: for
ADC: and ADC: only,
control bits

SDSTS

RETURNS +1: always

Currently a NOP for devices without device dependent statuses.

Gives illegal instructon PSI on the following errors:

DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal
DESX5: not open

TERMINAL SPECIFIC

INTRODUCTION

The JSYS's in this sub-section are currently implemented for use with terminals only; they have no effect or return default-value information when used with other devices.

Most JSYS's in this section are specified as giving an error if the device referenced is assigned to another job. However, it is legal to reference a terminal assigned to another job (as controlling terminal or with ASND) if the process executing the JSYS has the WHEEL special capability enabled.

JFN Mode Word

Each terminal in TENEX has an associated mode word of the following form. It is expected that such a word will also be associated with each JFN . This word can be read with RFMOD and changed with SFMOD and STPAR (all in this section).

A JFN which identifies a terminal (obtained from TTYn:) and which is opened in binary mode does not affect the terminal data mode of that terminal line, but instead performs binary input/output without reference to the terminal mode word. To open a terminal JFN in binary, the byte size should be 8, and the data mode should be Ø (default) or binary. A byte size of 8 with other data modes, or binary data mode with a byte size other than 8 are illegal.

bit	function
Ø:	output active (cannot be changed by SFMOD)
1:	has mechanical form feed
2:	has mechanical tab
3:	has lower case
4-1Ø:	page length
11-17:	page width
18-23:	wakeup control
	on:
	18: -
	19: -
	2Ø: formatting controls (1Ø,11,12,14,15,33,37)
	21: non-formatting controls
	22: punctuation char 4Ø-177 except those covered by bit 23
	23: alphanumeric char Ø-9 capital A - Z, lower case a - z
24-25:	echo mode
	ØØ: no echo
	Ø1: immediate echo only
	1Ø: immediate or deferred
	11: immediate and deferred
26:	accept links if 1
27:	allow advice if 1
28-29:	terminal data mode
	ØØ: "binary" 8-bit in and out; no format control or control group translation. No echoing.
	Ø1: "ASCII" - 7-bit in and out, plus parity on control group output. Format control and simulation, translation of control group on in (echo) and out as per SFCOC words.
	1Ø: Same as Ø1 except translation of control group for echoing disabled.

11: Same as Ø1 except translation of control group for output disabled.

3Ø: (lco) lower case output control
Ø: do not indicate
1: indicate by %X

31: (lci) lower case input control
Ø: no conversion
1: convert lower to upper

32-33: Full/Half duplex control
ØØ - Full duplex
1Ø - Character half duplex
11 - Line half duplex
Ø1 - Reserved

34: repeat last character (set by BKJFN, not affected by SFMOD)

35: If line is a dataset, on if "carrier is on"
If line is not a dataset, unspecified

Control Character Output Control

Each terminal has two control character output control (CCOC) words, consisting of 2-bit bytes, one for each of the characters Ø-37, interpreted as follows:

ØØ: IGNORE (send nothing)
Ø1: indicate by ↑X
1Ø: send actual code and account format action
11: simulate format action and account

The specific bytes for each character are given in the character set table (below); these words can be manipulated with RFCOC and SFCOC, described in this section.

TERMINAL CHARACTERISTICS CONTROL

TENEX provides capabilities for adjusting the behavior of the terminal output processing to suit various types of terminals. The characteristics which can be matched are:

1. mechanical form-feed, and tab present/absent on a device.
2. lower case available on a device
3. padding after carriage-return needed
4. padding after line-feed needed
5. padding after mechanical tab needed
6. padding after mechanical form-feed needed

Additional characteristics will be provided for as devices appear needing them.

Rather than provide for the independent setting of each of these parameters for each line (which would be costly in terms of resident storage), a number of known terminals have been identified, and each of the above parameters for each terminal has been specified. The terminal type may be specified for each line, and that specification implies the appropriate set of the above parameters. JSYS's have been implemented to allow reading and setting the terminal type. The following terminal type number specifications have been made:

Number	Terminal	Characteristics
Ø	TTY mod 33	no mechanical format, no padding
1	TTY mod 35	mechanical FF and TAB, pad TAB, pad CR
2	TTY mod 37	no mechanical format, has lower case
3	TI/EXECUPORT	has lower case, pad CR

Character Set

The following table gives information pertinent to the JSYS's in this section about each character in the TENEX character set. The "class" (used for wakeup control, see JFN Mode Word), is abbreviated as follows:

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

term. code dec.	ASCII code oct.	class	control character output control word-bits	character/↑equivalent
Ø	Ø	C	1-BØ,1	↑@ null, break
1	1	C	1-B2,3	↑A
2	2	C	1-B4,5	↑B
3	3	C	1-B6,7	↑C
4	4	C	1-B8,9	↑D
5	5	C	1-B1Ø,11	↑E
6	6	C	1-B12,13	↑F
7	7	C	1-B14,15	↑G bell
8	1Ø	F	1-B16,17	↑H backspace
9	11	F	1-B18,19	↑I tab
1Ø	12	F	1-B2Ø,21	↑J line feed
11	13	C	1-B22,23	↑K vertical tab
12	14	F	1-B24,25	↑L form feed
13	15	F	1-B26,27	↑M carriage return
14	16	C	1-B28,29	↑N
15	17	C	1-B3Ø,31	↑O
16	2Ø	C	1-B32,33	↑P
17	21	C	1-B34,35	↑Q
18	22	C	2-BØ,1	↑R
19	23	C	2-B2,3	↑S
2Ø	24	C	2-B4,5	↑T
21	25	C	2-B6,7	↑U
22	26	C	2-B8,9	↑V
23	27	C	2-B1Ø,11	↑W
24	3Ø	C	2-B12,13	↑X
25	31	C	2-B14,15	↑Y
26	32	C	2-B16,17	↑Z
27	33	F and C	2-B18,19	↑[esc, alt mode
	34	C	2-B2Ø,21	↑\
	35	C	2-B22,23	↑]
	36	C	2-B24,25	↑↑
	37	F	2-B26,27	↑← EOL end-of-line

4Ø	P	space
41	P	!
42	P	"
43	P	#
44	P	$
45	P	%
46	P	&
47	P	'
5Ø	P	(
51	P	)
52	P	*
53	P	+
54	P	,
55	P	-
56	P	.
57	P	/
6Ø-71	A	Ø-9
72	P	:
73	P	;
74	P	<
75	P	=
76	P	>
77	P	?
1ØØ	P	@
1Ø1-132	A	upper case letters A-Z
133	P	[
134	P	
135	P	]
136	P	↑ (carat on some terminals)
137	P	← (underline on some terminals)
14Ø	P	accent grave
141-172	A	lower case letters a-z
173	P	left curly brace
174	P	vertical bar
175*	P	right curly brace
176*	P	tilde
177	P	rubout

---

* If terminal does not have lower case (JFN mode word B3 off), 175 and 176 are converted to 33 on input, and treated accordingly.

CFIBF JSYS 100

Clears designated file input buffer

ACCEPTS IN 1: TENEX file designator

CFIBF

RETURNS +1: always

Acts like a NOP if designator associated with non-terminal.
Generates illegal instruction PSI on error conditions below.

CFIBF ERROR MNEMONICS:

DESX1:	illegal TENEX source/destination designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

CFOBF JSYS 101

Clears designated file output buffer

ACCEPTS IN 1: TENEX file designator

CFOBF

RETURNS +1: always

Acts like a NOP if designator associated with non-terminal.
Generates illegal instruction PSI on error conditons below.

CFOBF ERROR MNEMONICS:

DESX1:	illegal TENEX source/destination designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

SIBE JSYS 1Ø2

Skips if designated file input buffer empty

ACCEPTS IN 1: TENEX file designator

SIBE

RETURNS +1: with number of bytes now in input buffer in 2

+2: input buffer empty,
transparent to all AC's

Generates illegal instruction PSI on error conditions below.
Does not skip if designator associated with non-terminal.

SIBE ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

SOBE JSYS 1Ø3

Skips if designated file output buffer empty

ACCEPTS IN 1: TENEX file designator

SOBE

RETURNS +1: with number of bytes now in
output buffer in 2

+2: output buffer empty,
transparent to all AC's

Generates illegal instruction PSI on error conditions below
Always skips if designator associated with non-terminal.

SOBE ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

SOBF JSYS 175

Skips if designated file output buffer is full

ACCEPTS IN 1: TENEX file designator

SOBF

RETURNS +1: output buffer not full

+2: output buffer full

On either return, the number of bytes in the output buffer is returned in 2.

SOBF ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

DIBE JSYS 212

Dismiss until designated file input buffer is empty.

ACCEPTS IN 1: TENEX file designator

DIBE

RETURNS +1: always

If designator is associated with non-terminal, returns when end of file condition appears or file is closed.
Generates illegal instruction PSI on error conditions below.

DIBE ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

DOBE JSYS 1Ø4

Dismiss until designated file output buffer is empty

ACCEPTS IN 1: TENEX file designator

DOBE

RETURNS +1: always

Generates illegal instruction PSI on error conditions below. Returns immediately if designator associated with non-terminal.

DOBE ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

GTABS JSYS 1Ø5

Get tab settings for a particular file

ACCEPTS IN 1: TENEX file designator

GTABS

RETURNS +1: always with tab settings in accumulators 2,3,4

These accumulators are interpreted as a 1Ø7-bit bit string, with BØ of AC2 ignored, and each remaining bit indicating the presence of a tab in the corresponding column.

B1 of AC2 for column 1
BØ of AC3 for column 36
BØ of AC4 for column 72

Returns tab every 8 places if designator associated with non-terminal.

Generates illegal instruction PSI on error conditions below

GTABS ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

STABS JSYS 1Ø6

Sets tabs for a particular file

ACCEPTS IN 1: TENEX file designator

2,3,4: 3 words worth of tab setting bits, same format as GTABS

STABS

RETURNS +1: always

Acts like a NOP if designator associated with non-terminal. Generates illegal instruction PSI on error conditions below.

STABS ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

RFMOD JSYS 1Ø7

Reads a file's mode

ACCEPTS IN 1: TENEX file designator

RFMOD

RETURNS +1: always, mode word in 2

Returns ↑D66B1Ø+↑D72B17+7B3+actual data mode bits in 2 if designator associated with non-terminal. (See this section - JFN Mode Word.)

Generates illegal instruction PSI on error conditions below.

RFMOD ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

SFMOD JSYS 11Ø

Sets a particular file's modes

ACCEPTS IN 1: TENEX file designator

2: mode word

SFMOD

RETURNS +1: always

The fields of the JFN mode word affected by SFMOD are:

B18-23, wakeup control
B24-25, echo mode
B28-29, terminal data mode
B31 convert case on input

Acts like a NOP if designator associated with non-terminal.
Generates illegal instruction PSI on error conditions below.

SFMOD ERROR MNEMONICS:

DESX1: illegal value for designator
DESX3: no name for this JFN
DESX5: not open
DESX6: string pointer not legal
DEVX2: specified device assigned to another job

STPAR JSYS 217

Sets the terminal parameters for a particular terminal line.

ACCEPTS IN 1: TENEX file designator which must designate a terminal.

2: mode word

STPAR

RETURNS +1 always.

The fields of the JFN mode word affected by STPAR are:

B1,2,3 - mechanical characteristics: form feed, tab, lower case
B4-1Ø - page length
B11-17 - page width
B3Ø - convert case on output
B32-33 - duplex mode

Generates illegal instruction PSI if given non-terminal designator.

The following JSYS's deal with the terminal type number.

STTYP JSYS 3Ø2

Sets the terminal type number for a terminal line.

ACCEPTS IN 1: TENEX file designator which must identify a terminal line.

2: a terminal type number

STTYP

RETURNS +1: always

The terminal type number is set for the specified line. The three bits specifying mechanical tab, mechanical form-feed, and lower case are loaded into the JFN mode word from the device characteristics table, and they may be subsequently changed by STPAR if desired. An illegal instruction PSI is generated on error conditions below.

STTYP ERROR MNEMONICS:

DESX1: designator does not identify terminal line
STYPX1: terminal type number out of legal range

GTTYP JSYS 3Ø3

Gets the terminal type number for a terminal line.

ACCEPTS IN 1: TENEX file designator which must identify a terminal line.

GTTYP

RETURNS +1: always, with terminal type number in 2, and buffer allocation numbers in 3 (LH: number of buffers allocated on input, RH: number of buffers allocated on output). Generates an illegal instruction PSI on error conditions below.

GTTYP ERROR MNEMONICS:

DESX1: designator does not identify terminal line.

RFPOS JSYS 111

Reads a particular file's position

ACCEPTS IN 1: TENEX file designator

RFPOS

RETURNS +1: always, position in 2
bits Ø-17: position within a page (i.e., line number). Line number set to zero only when ↑L (FF) typed, and upon LOGIN.
bits 18-35: position within a line (i.e., column number)

Generates illegal instruction PSI on error conditions below. Returns Ø in 2 if designator associated with non-terminal.

RFPOS ERROR MNEMONICS:

DESX1: illegal value for designator
DESX3: no name for this JFN
DESX5: not open
DESX6: string pointer not legal
DEVX2: specified device assigned to another job

RFCOC JSYS 112

Reads a file's control character output control

ACCEPTS IN 1: TENEX file designator

RFCOC

RETURNS +1: always,output control words in 2 and 3

Returns 525252525252 in 2 and 3 if designator associated with non-terminal.

Generates illegal instruction PSI on error conditions below.

RFCOC ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

SFCOC JSYS 113

Sets a file's control character output control

ACCEPTS IN 1: TENEX file designation

2,3: output control words

SFCOC

RETURNS +1: always

Generates illegal instruction PSI on error conditions below. Acts like NOP if designator associated with non-terminal.

SFCOC ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX3:	no name for this JFN
DESX5:	not open
DESX6:	string pointer not legal
DEVX2:	specified device assigned to another job

STI JSYS 114

Simulate terminal input

ACCEPTS IN 1: TENEX file designator (only terminal designators are legal)

2: the character (right justified) to be input

STI

RETURNS +1: always

The character is put into the input buffer whether or not the input buffer is empty. DIBE can be used to prevent sending an interrupt character (e.g., ↑C) before the program has processed all of the previous input.

Generates illegal instruction PSI on error conditions below

STI ERROR MNEMONICS:

TTYX1:	a non-terminal designator or non-existing terminal was specified
DEVX2:	specified device assigned to another job

DTACH JSYS 115

Detaches the controlling terminal from a job

DTACH

RETURNS +1: always

Acts like a NOP if job is already detached. Appends console detached entry to FACT accounting file on actual detach.

ATACH JSYS 116

Attaches the controlling terminal to a job

ACCEPTS IN 1: the tss job number of the designated job

2: left half:
BØ off, when the job gets attached it simply continues running
BØ on, when the job gets attached, a ↑C is faked
NOT IMPLEMENTED YET

right half:
the directory number under which the job to be attached is logged in.

3: a string pointer to the beginning of an ASCIZ password string in the address space of caller

ATACH

RETURNS +1: unsuccessful, error # in 1

+2: successful, If job executing ATACH was logged in, this job is now detached with primary I/O not redirected; hence, if a process had primary I/O from the controlling terminal, it will hang when it attempts primary I/O.
If job executing ATACH was not logged in and ATACH is successful, the job quietly goes away. ATACH appends a console attached entry to the FACT accounting file.

It is normally illegal to ATACH to a job which already has a controlling terminal. However, if the process executing ATACH has the WHEEL or OPERATOR special capability enabled, it can ATACH to a job regardless of whether it has a controlling terminal and whether or not the directory and password given are correct.

ATACH ERROR MNEMONICS:

ATACX1: illegal job number
ATACX2: job already attached to a controlling terminal
ATACX3: directory number does not match
ATACX4: password incorrect
ATACX5: job executing ATACH has no controlling terminal

Terminal Linking

It is possible to link the output of any line to up to four other lines. A linked line gets exactly what the source line gets; the linked line has no effect on the formatting of the source line. (In particular, a linked line can overstrike at the right margin because the combined string from the link and from the controlling job can total more than 72 printing characters, without a carriage return. Note also that if the source line has mechanical tab, the linked line will also be sent the tab whether or not it also has mechanical tab.)

Internally, linking is accomplished by placing the desired 'remote' line number in any of the four 9-bit fields of word TTLINK(SOURCE). The accept/refuse message bit is defined to be bit 26 in the terminal (JFN) mode word. It can be read by RFMOD, but cannot be set by SFMOD (for backward compatibility reasons). Rather, it is set by one of the options of TLINK described below.

TLINK JSYS 216

Terminal link control

ACCEPTS IN 1: XWD function bits, object designator
2: remote designator

TLINK

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

The object and remote designators are standard file designators. However, only 4xxxxx and 77777 are allowed. The function bits are executed from left to right (Ø to 17) and organized in what should be the most useful order.

BØ clear link from remote (all if -1) to object
B1 clear link to remote (all if -1) from object

B2 establish link to remote from object
B3 establish link from remote to object

B4 set accept bit for object to state specified by bit 5
B5 desired state of accept bit if bit 4 on

Restrictions if not WHEEL:

Object designator must specify TTY assigned to this job.

'object-to-remote' must be done before 'remote-to-object' (but can be done in same instruction)

If accept bit of remote is off, 'object-to-remote' will cause ringing of remote bell; if remote does not set accept in 15 seconds, TLINK will return failure.

TLINK ERROR MNEMONICS:

DESX1: illegal designator
TLNKX1: attempt to set remote to object before object to remote

ADVIZ JSYS 315

Sets up terminal lines useful for the ADVIZing function.

ACCEPTS IN 1: LH:
B0 clear ADVIZ link
B1 set to ADVIZ line given in right half
B2 set to receive advice from designated line in RH

RH: terminal line designator

RETURNS +1: error, error code in AC1
+2: Success

ADVIZ

An ADVIZ link is established by having the advisee execute a "receive advice" (B2) designating the intended advisor, and then the advisor executing a "send advice" (B1) designating the advisee line. An ADVIZ link causes the advisor's characters to be input to the advisee's job. ADVIZ provides terminal output using the TLINK mechanism. While ADVIZ is in progress, a ↑C character typed by the advisor causes the ADVIZ link to be broken, but leaves the output link in effect. A ↑C will be sent to the advisee job when the advisor types a ↑Y.

ADVIZ ERROR MNEMONICS:

ADVX1: advice refused
ADVX2: not used
ADVX3: B1 and B2 of AC1 both on in call
ADVX4: an attempt to accept advice with ADVIZ already in progress

TLINK errors can also occur.

ASNDP JSYS 26Ø

Assign Display Process

ASNDP

RETURNS +1: error, no display processes available

+2: success with the assigned display process number in AC1

RELDP JSYS 261

Release Display Process

ACCEPTS IN 1: a display process number or -1 to release all display processes assigned to this job

RELDP

RETURNS +1: always, the indicated display processes are deassigned if they had been assigned. If the processes were running, they are first stopped with STPDP.

ASNDC JSYS 262

Assign Display Console

ACCEPTS IN 1: a mask of ones for each display console to be assigned. BØ => console Ø etc.

ASNDC

RETURNS +1: error, cannot assign one or more of the requested consoles, no consoles are assigned.

+2: success

RELDC JSYS 263

Release Display Console

ACCEPTS IN 1: a mask of ones for each display console to be deassigned

RELDC

RETURNS +1: always. An attempt to release an unassigned console is treated as a NOP.

STRDP JSYS 264

Start Display Process

ACCEPTS IN 1: BØ if one, set the memory map

B1 if one, set the frame iteration count

B2 if one, set the console mask

B4 if one, set the interrupt channel assignment words

B5 if one, set the contents of the display processor registers

RH: a display process number.

2: location of a table to set the initial status of the display process.
Only used if LH(1) is non-zero.

STRDP

RETURNS +1: error

+2: success. If the process was running, it is stopped first.

Status Table Format

map word	negative count of pages,,first page (maximum of 1ØØ(8) pages)
frame iteration count	if Ø, indefinite iteration if >Ø, an iteration count
console mask CONI word interrupt channel assignments	three words of 6 bit bytes, one byte for each type of display interrupt; byte contains channel number for pseudo-interrupt or 77 if not assigned
interrupt conditions	

byte	condition
Ø	program stop with C(P1) >1ØØ program stop with C(P1) = Ø is a frame stop
1	stop on hit
2	frame, iteration count completed
3	I/O error
4	memory-to-memory stop (WCR positive)
5	program stop with C(P1) =1 (display process continued automatically)

display registers the order of the registers is:

WCR
RCR
UR
RAR
WAR
DSP
P1
DIR
PC
RSR
SR
SP
SAVELB
SAVERT
VIEWLB
VIEWRT
WINDLB
WINDRT
INSTLB
INSTRT
NAME
CDIR
HITANG
SELINT

STRDP ERROR MNEMONICS:

DPX1:	display process number not assigned to this job
DPX2:	illegal display process number
STRDX1:	invalid map word
STRDX2:	can not lock as many pages as specified
STRDX3:	page without read, write (or write copy) and execute access

STPDP JSYS 265

Stop Display Process

ACCEPTS IN 1: a display process number

STPDP

RETURNS +1: always. This is a NOP for non-running display processes or for processes assigned to other jobs.

STSDP JSYS 266

Set Status of Display Process

ACCEPTS IN 1: BØ if one, set the memory map

B1 if one, set the frame iteration count

B2 if one, set the console mask

B4 if one, set the interrupt channel assignment words

B5 if one, set the contents of the display processor registers

RH: a display process number

2: the location of the table from which to set the status of the indicated display process.

STSDP

RETURNS +1: error

+2: Success. If the process is running, it is first stopped and then restarted after its status is changed.

STSDP is the same as STRDP if the display process is running.

STS IP ERROR MNEMONICS:

DPX1:	display process number not assigned to this job
DPX2:	illegal display process number
STRDX1:	invalid map word
STRDX2:	can not lock as many pages as specified
STRDX3 :	page without read, write (or write copy) and execute access

RDSDP JSYS 267

Read Status of a Display Process

ACCEPTS IN 1 : a display process number

2: location where to store the status information

RDSDP

RETURNS +1 : error

+2 : Success. The status information is stored in the same format as specified for STRDP and STSDP.

RDSDP ERROR MNEMONICS:

DPX1:	display process number not assigned to this job
DPX2:	illegal display process number

WATDP JSYS 27Ø

Wait for Display Process to Stop

ACCEPTS IN 1: a display process number

WATDP

RETURNS +1: always, after waiting for the display process to stop

OTHER DEVICES

DVCHR JSYS 117

Get device characteristics.

ACCEPTS IN 1: JFN
or device designator

DVCHR

RETURNS +1: device designator in 1 (even if JFN was supplied to call)
device characteristics in 2
device unit number in right half of 3
TSS job # to which device is assigned (if it is) in left half of 3

characteristics

BØ device can do output
B1 device can do input
B2 device has a directory
B3 device is assignable with ASND
B4 device has multiple directories
B5 device available or assigned to this job
B6 device is assigned by ASND
B7 device is mountable
B8 device is mounted
B9-B17 device type
B2Ø-B35 if on, mode is legal for mode 35-n

Generates illegal instruction PSI on error conditions below.

DVCHR ERROR MNEMONICS:

DEVX1: illegal device designator
DESX1: illegal value for designator
DESX3: no name for this JFN
DESX4: terminal designator or string pointer not legal

STDEV JSYS 12Ø

String to device designator.

ACCEPTS IN 1: a string pointer to string of form
DTA1(terminator)

STDEV

RETURNS +1: unsuccessful, error # in 2

+2: successful, device designator in 2

STDEV ERROR MNEMONICS:

STDVX1: unrecognized device

DEVST JSYS 121

Device designator to string.

ACCEPTS IN 1: a TENEX destination designator

2: a device designator

DEVST

RETURNS +1: unsuccessful, error # in 2

+2: successful, with updated string pointer in 1
(if pertinent)

DEVST ERROR MNEMONICS:

DEVX1: illegal device designator
DESX1: illegal value for designator
DESX2: terminal not available to this job
DESX3: no name for this JFN
DESX5: not open

MOUNT JSYS 122

Mounts a device. Mountable devices such as DECtapes must be MOUNTed before being accessed.

ACCEPTS IN 1: device designator
(if B3 on do not read directory, off read directory)

MOUNT

RETURNS +1: unsuccessful, error # in 1

+2: successful

If device is already mounted, it is first dismounted.
If B3 of 1 is on, a DECtape or magtape is assumed to be non-directory.

MOUNT ERROR MNEMONICS:

DEVX1:	illegal device
DEVX2:	device not available
MNTX1:	illegal directory format and directory read specified
MNTX2:	failed to mount (e.g. device off-line)
MNTX3:	device type not mountable
DSMX1:	failed to dismount (see DSMNT)

I/O errors can also occur

DSMNT JSYS 123

Dismount a device.

ACCEPTS IN 1: device designator

DSMNT

RETURNS +1: unsuccessful, error # in 1

+2: successful, directory is updated if necessary

DSMNT ERROR MNEMONICS:

DEVX1:	illegal device
DEVX3:	device wasn't mounted
DSMX1:	cannot dismount (e.g. files open on device)

I/O errors can also occur

INIDR JSYS 124

Initializes the directory of a mounted device.

ACCEPTS IN 1: device designator

INIDR

RETURNS +1: unsuccessful, error # in 1

+2: successful, directory initialized

INIDR ERROR MNEMONICS:

DEVX1:	illegal device
DEVX3:	device wasn't mounted
INIDX1:	device busy

I/O errors can also occur

RDDIR JSYS 32

Read Device DIRectory

ACCEPTS IN 1: device designator
(only DECtapes currently allowed)

2: core address

RETURNS +1: unsuccessful, error code in 1

+2: directory has been copied to specified address in caller's address space

RDDIR ERROR MNEMONICS:

RDDIX1: cannot read directory for this device
(not DECtape, not mounted, etc.)

GDSKC JSYS 214

Get DiSK Count. JSYS to determine file storage usage.

ACCEPTS IN 1: device designator
(currently only 777777 for DSK: allowed)

GDSKC

RETURNS +1: 1: number of pages in use
2: number of unused pages

FDFRE JSYS 213

Determines File Directory FREe space.

ACCEPTS IN 1: TENEX device designator
(only DSK legal at present)

2: directory number

FDFRE

RETURNS +1: with number in 2 representing amount of free space left

For disk directories, the amount of free space left is returned as a number of words.

Generates illegal instruction PSI on error conditions below.

FDFRE ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	terminal not available to this job
FDFRX1:	not a multiple directory device (at present, not DSK)
FDFRX2:	no such directory number

LITES JSYS 215

Displays data in the MI lights on the central processor console.

ACCEPTS IN 1: 36-bit word to be displayed

LITES

RETURNS +1: successful, generates illegal instruction PSI on error conditions listed below.

LITES ERROR MNEMONICS:

WHELX1:	WHEEL, OPERATOR or MAINTENANCE special capability not enabled

Section 4 Index

PSEUDO-INTERRUPT JSYS'S

INTRODUCTION

This section of the manual describes the JSYS's for controlling the pseudo-interrupt system. The pseudo-interrupt system provides a means by which certain conditions cause control to be transferred to specified locations in the process called the pseudo-interrupt routines. Simultaneously, the process PC will be saved so that the pseudo-interrupt routine may resume the interrupted sequence upon completion of its tasks. These conditions are:

(1) Terminal Pseudo Interrupts--generated when selected terminal keys are typed.
(2) Illegal Instruction Traps (such as attempts to execute I/O instructions in ordinary user mode) or attempts to execute privileged monitor calls.
(3) Memory Traps including read, write and execute, directed traps, and un-assigned memory.
(4) Arithmetic Processor Traps
(5) Unusual File Conditions (EOF, errors)
(6) Specific Time of Day reached
(7) Generated Pseudo-Interrupts
(8) Subsidiary Fork Termination
(9) System Resource Allocation traps

There are 36 pseudo-interrupt channels and some number NPLEVS (currently 3), pseudo-interrupt priority levels. Priority levels are numbered from 1 to NPLEV. Ø is not a legal priority level. Most of the possible causes of PSI's are each permanently assigned to one of the 36 channels. The remainder are user-assignable to one of several of the 36 channels. Each channel can be activated or deactivated. If activated, the channel can cause an interrupt on the user-specified priority level. This two step interrupting procedure eliminates the need for user decoding of interrupt cause.

An interrupt for any channel will be initiated only if there are no interrupts in progress on the same or higher priority channels. Otherwise, it will be remembered and initiated when the last equal or higher priority channel de-breaks. Since a higher level (lower priority number) interrupt can interrupt a lower level PSI routine, there can be up to NPLEVS interrupts in progress simultaneously. The user's PSI routine exits with a monitor call which resets the interrupt-in-progress status of that PSI priority level.

The user can turn the pseudo-interrupt system on or off. When the system is off, interrupt requests are remembered until the system is turned back on except for certain "panic" channels (e.g., instruction trap, described below) where an interrupt request will take as though the PSI system was always on. The user can also clear the entire interrupt system, thereby forgetting all stacked requests.

Channels vs. Priority Levels

Note carefully the distinction between channels and priority levels. A channel corresponds to one particular cause of interrupts. There is a one-to-one correspondence of channels and interrupt causes as shown in Table 1. Some interrupt causes are assignable to different channels, but at any given time, an interrupt cause is associated with at most one channel, and each channel is associated with at most one interrupt cause. The priority levels are provided to allow some interrupt conditions to be able to interrupt the service routine for other interrupt conditions.

For example, a program could assign a "break key" type of user request (e.g., LISP's control-H) to a low priority level, some APR overflow conditions to a medium priority level, and an "abort key" user request (e.g., LISP's control-E) to a higher priority channel. Thus, an unexpected or infrequently occurring APR condition (e.g., PDL overflow) could be handled during the program's main sequence or during a user initiated break sequence, but a user initiated abort request would override any of the above sequences.

As stated earlier, each activated PSI channel is user assigned to one of NPLEVS priority levels. The user makes his assignment by considering when one interrupt condition can arrive during the servicing of another, and when servicing of the later interrupt cannot be deferred until completion of the first. Note that the pseudo-interrupt levels are numerically referenced as 1, 2, and 3. Level Ø is not legal and in particular if a pseudo-interrupt request occurs in a fork where the level associated with that channel is Ø, the system considers the fork not prepared to handle the interrupt, and freezes or terminates it according to bit 17 of the capabilities word.

Interrupt Service Conventions

Before using the pseudo-interrupt system, the user must execute a monitor call to specify the location of his channel table (CHNTAB) and level table (LEVTAB) in two half-words, i.e.

```
LEVTAB,,CHNTAB
```

which the monitor will keep in the PSB. For each channel to be activated, the user must set up the contents of location CHNTAB plus the channel number to contain:

Left half: number of priority level to which this channel is assigned.
Right half: address of interrupt service routine for this channel.

For any priority level specified by one or more of the above channel words, the user must set up the contents of location LEVTAB plus the priority number minus 1 to contain:

Left half: (unused)

Right half: location of word (in writable page) in which to store interrupt PC and flags.

When an interrupt is requested, the channel word (at location CHNTAB plus the interrupt channel number) is fetched. The left half specifies the priority level to be used. If this left half is Ø, or if the pseudo interrupt system for this fork is off or if an SIR has never been done for this fork, the system considers this fork is not prepared to handle a pseudo-interrupt on this channel and the pseudo-interrupt is changed to a fork terminating condition; otherwise if neither that level nor any higher priority level interrupt is in progress, the process PC will be set to the right half of the channel word. The old process PC will be stored as specified by the right half of the priority level word (at location LEVTAB plus the priority number minus 1), and the process will be run. When the interrupt routine is completed, it should be dismissed with a monitor call which restores the process PC as specified by the right half of the priority level word, and the process is resumed.

There are some special conditions governing interrupts from monitor calls. However, if the service routine does not change the interrupt PC, all interrupts are guaranteed to be completely transparent, i.e., the fork will be resumed on de-break and will continue to do whatever it was doing. Note that, in general, the service routine must save any AC's or other temporary storage possibly in use by the interrupted routine. The monitor protects temporary storage in use by interrupted monitor routines, but the user is responsible for protecting all temporary storage in user memory. If the service routine does change the interrupt PC, (in any way, even the flag bits) the de-break will cause the fork to be restarted at the location specified by the interrupt PC.

When a user program does a DEBRK, the monitor checks the contents of the return PC word to determine how to resume. If the PC word was not changed, the process will be restored to its state prior to the pseudo-interrupt. For example, if the process was dismissed waiting for I/O to complete, it will be restored to that state after the DEBRK. If the contents of the return PC word are modified, the DEBRK will cause the process to start running at the return PC location. The user can tell by looking at the User/Exec mode bit of the return PC word whether the process was in monitor code (executing a monitor call) or in user code when the pseudo-interrupt request was granted. That is, if B5 is Ø, the process was in monitor code; if B5 is 1, the process was in user code.

Channel Usage

Only channels Ø-5 and 24-35 may be used for terminal character interrupts. Many other channels are permanently assigned to various error conditions. Any channel may be used for program-initiated interrupts (IIC JSYS).

The details of special channel usage are summarized in the following table:

channel	note	special use
Ø-5		terminal character
6		APR overflow (includes NODIV)
7		APR Floating overflow (includes FXU)
8		not used
9	(1)	APR PDL overflow
1Ø		file condition 1, EOF
11	(1)	file condition 2, data error
12		file condition 3, (unassigned)
13		file condition 4, (unassigned)
14	(2)	time of day
15	(1)	illegal instruction
16	(1)	illegal memory read
17	(1)	illegal memory write
18	(1)	illegal memory execute
19		subsidiary fork termination or forced freeze
2Ø	(1,2)	machine size exceeded
21		trap to user (see SPACS)
22		reference to non-existent page
23		not used
24-35		terminal character

notes: (1) panic channel (see below)
(2) NOT IMPLEMENTED YET

Panic Channels

Certain channels including APR PDL overflow, file data error, illegal instruction, illegal memory read, illegal memory write, illegal memory execute, and machine size exceeded, etc. are special "panic" channels in that they cannot be completely turned off. While they will respond normally to the channel on/off and read channel mask JSYS's, a pseudo interrupt request received on such a channel which has been turned off will be considered a fork terminating condition.

Pseudo Interrupt Fork Specification

When a particular pseudo-interrupt condition arises, one fork will be pseudo-interrupted. It is often not obvious which fork should be interrupted. For example, when a terminal pseudo-interrupt character is typed, it is quite possible that several forks may be armed for that pseudo-interrupt condition. The following rules specify which fork gets the various pseudo-interrupts.

Terminal Pseudo-Interrupts

Up to 36 terminal keys may be used to specify pseudo-interrupts. Each of these may be armed in multiple forks. When a terminal interrupt occurs, the job's fork structure is searched top down, depth first, and the last active (not frozen or terminated) fork found will receive the interrupt. This is usually the deepest fork with the oldest branches.

When a fork is frozen, its terminal interrupts are effectively deactivated, although they will continue to be indicated in the terminal interrupt word (obtained from RTIW) for that fork. When the fork is resumed, the terminal interrupts will be automatically reactivated.

When any fork freeze or unfreeze operation is completed, or when any other operation is completed which explicitly changes the state of the terminal interrupt word for a fork, the terminal interrupt word for the job (and for the terminal line if the job is attached) is set to be the inclusive OR of all of the non-frozen forks in the job. When searching for a fork to interrupt after receipt of an interrupt character, frozen forks will not be considered. The characters which actually cause interrupts are determined by masking the inclusive-OR obtained above with a terminal interrupt enable mask. This mask may be set/read with STIW and RTIW with the whole job (-5) argument. This word is normally -1 and thus enables all terminal interrupts to be activated. A zero in any position of this word will prevent that terminal interrupt from being concurrently active, but the fact that a code has been activated will be remembered and the activation will take effect when the mask is changed so as to have a one in the corresponding position. The function of this mask is to allow programs to turn off interrupt codes activated by superior forks, for example control-C. Reading and changing this mask word requires the control-C capability to be enabled.

Directed Pseudo-Interrupts

The generated pseudo-interrupts are directed to specific fork(s) which completely specifies the fork to interrupt.

Terminating Conditions

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

1. the interrupt system for the fork is off, or
2. no SIR has been done, or
3. no level has been assigned to the channel (i.e. left half of channel's word in CHNTAB is Ø)
4. the channel has been "reserved" by the superior fork (see SIRCM).

then the interrupt is considered a forced termination condition. In this case the fork which was to have received the interrupt is halted or frozen according to bit 17 of the capabilities word, and a fork termination interrupt is sent to its superior (see RFSTS).

Program Conditions

Other conditions arising from program execution including non-error file conditions (such as EOF), inferior fork termination, and APR traps (overflow, floating overflow, floating underflow, and no-divide) are handled as for number 3 above except that the process continues in sequence if the channel is not armed. The monitor will set the actual APR bits in a manner appropriate to each process each time the monitor begins to run the process.

Fork Termination

When a fork terminates, only the immediate superior will be checked for fork termination interrupt enabled.

Terminal Interrupt Codes

There are 36 codes used to specify terminal characters or conditions on which terminal interrupts are to be initiated. A process can assign a character or condition to a specific PSI channel with the ATI JSYS, described in this section.

terminal code (decimal)	ASCII value (octal)	character or condition
Ø	Ø	↑@ or break
1	1	↑A
2	2	↑B
3	3	↑C
4	4	↑D
5	5	↑E
6	6	↑F
7	7	↑G
8	1Ø	↑H
9	11	↑I (tab)
1Ø	12	line feed
11	13	↑K (vertical tab)
12	14	↑L (form feed)
13	15	carriage return
14	16	↑N
15	17	↑O
16	2Ø	↑P
17	21	↑Q
18	22	↑R
19	23	↑S
2Ø	24	↑T
21	25	↑U
22	26	↑V
23	27	↑W
24	3Ø	↑X
25	31	↑Y
26	32	↑Z
27	33	esc (altmode)
28	177	rubout
29	4Ø	space
3Ø	---	dataset carrier off
31-35		unassigned

Deferred Terminal Interrupts

Deferred terminal interrupts are optionally available. Previously, terminal interrupts always took effect immediately, i.e., upon receipt by the terminal service routine. Now, any enabled terminal interrupt may be declared "deferred" by use of STIW as described below. When a character which has been declared to be a deferred interrupt character is received, it is placed in the input buffer in sequence with the other characters of the input stream. Only when the program reaches that point in the input stream and attempts to read the interrupt character does the interrupt action occur. The interrupt character itself is NOT passed to the program as a data character in the input stream.

The intent of this facility is to allow interrupt actions to occur in sequence with other actions specified by the input stream, when characters are typed ahead of the time that the programs request them. It is therefore important to understand the interaction of multiple forks in this situation. Note that the deferred interrupt occurs when any fork of the job executes an input JSYS (PBIN, BIN, etc.) which would remove the interrupt character from the input buffer. This fork may or may not be the one which is enabled for that interrupt and which will be interrupted. The following statements cover all of the cases:

a. If the fork attempting the read which causes the interrupt is also the fork which is to be interrupted, then it is guaranteed that the interrupt will occur before any more characters are passed to the program.

b. If the fork to be interrupted is the top fork (usually the EXEC), then the interrupt will occur before any more characters are passed to the program unless a second fork is also attempting to read from the same source (normally an anomalous condition).

c. If neither of these cases apply, then the fork doing terminal input will continue to run and may succeed in reading one or more characters before the interrupt can take effect. This is unavoidable since the input fork and the interrupted fork will logically be running in parallel.

An otherwise deferred interrupt character will have immediate effect if two of that character are received with no intervening characters. There is no element of time involved here, rather it is required only that the character now being received is the same as the last previous character received and that the character is a deferred

interrupt character. When this event occurs, the terminal input buffer is automatically cleared (simulated CFIBF). Note also that the system maintains a separate one-character buffer to detect this situation, so that it will work even if the input buffer is otherwise full.

Process Pseudo-Interrupt Tables

Before using the PSI system, a process must set up the following two tables, and declare their locations with SIR.

LEVTAB

Three words. Left halves unused, right half of word Ø points to word in process's address space in which to store the return PC word for level 1, right half of word 1 is for level 2, etc.

CHNTAB

Indexed by channel number. Left half of each word contains priority level (1, 2, or 3) for that channel. Right half contains address of PSI routine for that channel.

SIR JSYS 125

Sets up the pseudo-interrupt table addresses for a process

ACCEPTS IN 1: a fork handle

2: LH: the virtual address of LEVTAB
RH: the virtual address of CHNTAB

SIR

RETURNS +1: always, places the contents of accumulator 2 in a word in the PSB.

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

SIR ERROR MNEMONICS:

SIRX1:	address of LEVTAB or CHNTAB < 2Ø
FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks

RIR JSYS 144

Read pseudo-interrupt table addresses for a process, i.e., the channel and level table addresses as set by SIR.

ACCEPTS IN 1: a fork handle

RIR

RETURNS +1: always with addresses in 2:
LH: LEVTAB address
RH: CHNTAB address

Returns Ø if no SIR has been done for designated process. This JSYS allows several independent programs to reside in one fork and share interrupt tables.

Error conditions

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

EIR JSYS 126

Enables the pseudo-interrupt system for a process

ACCEPTS IN 1: a fork handle

EIR

RETURNS +1: always

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

EIR ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork

SKPIR JSYS 127

Tests the pseudo interrupt system to see if it is enabled i.e., skip if pseudo interrupt system on

ACCEPTS IN 1: a fork handle

SKPIR

RETURNS +1: pseudo interrupt system off
+2: pseudo interrupt system on

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

SKPIR ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

DIR JSYS 13Ø

Disables the pseudo-interrupt system for a process

ACCEPTS IN 1: a fork handle

DIR

RETURNS +1: always

If pseudo-interrupt requests are generated while the pseudo-interrupt system is off, they are remembered and will take when the pseudo-interrupt system is turned back on unless an intervening CIS is issued. Note that once the ↑C terminal key gets assigned to a channel, it will always cause a pseudo-interrupt which cannot be disabled by DIR but can be disabled by shutting off the channel in the fork assigned to ↑C or by the monitor during non-pseudo-interruptible periods. DIR has no effect on pseudo-interrupts on panic channels (i.e., they will behave as though the pseudo-interrupt system were enabled in that fork).

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

DIR ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork

AIC JSYS 131

Activate specified channels

ACCEPTS IN 1: a fork handle

2: a 36-bit word, BØ on means activate channel Ø, B1 on means activate channel 1,...

AIC

RETURNS +1: always

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

AIC ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks

IIC JSYS 132

Initiates pseudo-interrupts on specified channels in a fork

ACCEPTS IN 1: a fork handle

2: a 36-bit word, BØ on means initiate pseudo-interrupt on channel Ø, B1 on means initiate pseudo-interrupt on channel 1,...

IIC

RETURNS +1: always

Generates illegal instruction pseudo-interrupt on error conditions below.

IIC ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX3:	fork handle cannot reference multiple forks for this JSYS

DIC JSYS 133

Deactivate specified channels

ACCEPTS IN 1: a fork handle

2: a 36-bit word, BØ on means deactivate channel Ø, B1 on means deactivate channel 1,...

DIC

RETURNS +1: always

A deactivated channel completely ignores pseudo-interrupt requests made of that channel except for IIC JSYS's to that channel or some exceptions noted under DIR.

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

DIC ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks

RCM JSYS 134

Read channel-activated word-mask

ACCEPTS IN 1: a fork handle

RCM

RETURNS +1: always, 36-bit word in 1 with BØ on meaning channel Ø activated, B1 on meaning channel 1 activated,...

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

RCM ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks

RWM JSYS 135

Read Waiting Channel interrupts word mask

ACCEPTS IN 1: a fork handle

RWM

RETURNS +1: always, 36-bit word in 1 with BØ on meaning channel Ø interrupt waiting, B1 on meaning channel 1 interrupt waiting, etc. The status of the pseudo-interrupts in progress is returned in 2, where B1 means level 1 in progress, B2 means level 2 in progress, etc.

Generates an illegal instruction pseudo-interrupt on error conditions listed below.

RWM ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks

SIRCM JSYS 142

Set Inferior Reserved Channel Mask.

ACCEPTS IN 1: inferior fork handle

2: mask bit on for reserved channels

SIRCM

RETURNS +1: always

SIRCM must be given only for inferior forks. PSI channels with the corresponding mask bit on will never break to the inferior fork, but rather will cause termination or freezing. This provides a facility for a superior to monitor an inferior (e.g., illegal instructions, memory traps) regardless of how the inferior uses the interrupt system.

SIRCM ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

RIRCM JSYS 143

Read Inferior Reserved Channel Mask.

ACCEPTS IN 1: fork handle, inferior or self

RIRCM

RETURNS +1: always with
2: reserved channel mask for designated fork

Note that a fork may read but not set its own reserved channel mask.

RIRCM ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

DEBRK JSYS 136

Dismiss current pseudo-interrupt level in progress and resume process in sequence, i.e., debreak

DEBRK

Normally returns to the location specified by the PC stored in the priority level PC word.

RETURNS +1: if no pseudo-interrupt is currently in progress

ATI JSYS 137

Assigns a terminal code to a channel

ACCEPTS IN 1: LH: terminal code
NOT an ASCII code; see introduction
RH: channel number

ATI

RETURNS +1: always

If there is no controlling terminal (i.e., the job is detached), the assignments are remembered and will be in effect when a terminal becomes attached. ATI also sets the corresponding bit in the fork's terminal interrupt mask (see STIW/RTIW).

Generates an illegal instruction pseudo-interrupt on any error conditions shown below.

ATI ERROR MNEMONICS:

TERMX1: illegal to associate this particular terminal code with pseudo-interrupt channel

ATIX1: illegal channel number for terminal code assignment (only Ø-5, 24-35 legal)

ATIX2: attempt to assign ↑C but this special privilege not enabled for this process

DTI JSYS 14Ø

Deassigns terminal code

ACCEPTS IN 1: terminal code
NOT an ASCII code; see introduction

DTI

RETURNS +1: always

Generates an illegal instruction pseudo-interrupt on any error conditions shown below

DTI ERROR MNEMONICS:

TERMX1: illegal to associate this particular terminal code with a pseudo-interrupt channel

DTIX1: this terminal code was never assigned to this process

STIW JSYS 174

Set Terminal Interrupt Word. This JSYS sets the mask word for any specified fork or the whole job, thus determining whether that fork/job will receive an interrupt for each of the 36 terminal interrupt codes. If the bit in the job mask for a particular terminal code is off, that character will remain in the input stream. If the bit is on, the character will not be placed in the input stream, and the fork structure will be searched for forks having the code enabled. The lowest such fork found will be interrupted. If no fork is found, the bit is cleared from the job mask and the character is lost.

Each STIW done on a specific fork will cause the job mask to be updated for the bits changed in that fork.

Note that STIW can turn particular terminal codes on and off for a particular fork without changing the interrupt channel assignment that each code may have. ATI must be used to set the channel assignment.

ACCEPTS IN 1: BØ on, use 3 as below
BØ off, ignore 3
RH: a fork handle or -5 for whole job

2: Mask for terminal codes, bit N a 1 means terminal code N enabled

3: Deferred interrupt mask
(ignored unless BØ of 1 is on
and RH of 1 names a specific fork)

STIW

RETURNS +1: always

A 1 in any bit of the deferred interrupt mask causes the corresponding character to be deferred. If more than one fork has enabled a particular interrupt character, it will be deferred if ANY of the forks have declared it to be deferred. The AC3 argument is ignored, and no change is made to the deferred interrupt mask, if the sign bit of AC1 is off or if the right half of AC1 is not a specific fork designator.

STIW ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

RTIW JSYS 173

Read Terminal Interrupt Word. Reads mask word for any specified fork or whole job.

ACCEPTS IN 1: BØ on: return AC3
BØ off: do not change AC3
RH: a fork handle or -5 for whole job

RTIW

RETURNS +1: always, with 2: terminal interrupt mask

3: Deferred interrupt mask
if BØ of 1 is on and RH of 1 specifies a particular fork (not -5)

RTIW ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

CIS JSYS 141

Clear Interrupt system. Clears all breaks in process and waiting breaks.

CIS

RETURNS +1: always

RWSET JSYS 176

Release Working Set

Removes all of a process's pages from its working set. Operation is invisible to the user, but allows the pages to move to secondary storage; they will not be preloaded the next time the process is swapped in.

RWSET

RETURNS +1: always

GTRPW JSYS 171

Get trap words. This JSYS allows a program to retrieve information about a previous memory read, write or XCT trap.

ACCEPTS IN 1: fork handle

GTRPW

RETURNS +1: always, with:
1: Trap status word from last memory trap
2: Write data (if any)

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.

If there have been no memory traps, Ø is returned in 1.

Pager Traps

A paging trap will occur whenever one of the following events happens:

(1) The trap bits in the process page table force a trap.

(2) The addressed page (or indirecting page table) is not in core.

(3) An illegal condition is detected.

(4) The Core Status Table entry for an addressed page contains an age with the three highest bits = Ø.

When one of the above conditions happens, the pager first stores the cause and location of the trap into the P.S.B. at location 571 octal. The format of this word is shown below. Then, if the APR operation in progress was a write, it stores the data into the P.S.B. Finally, it forces the APR to execute the trap routine.

The arrangement of the Trap Cause field was chosen so that decoding of the cause could be easily accomplished by the JFFO instruction.

If the trap word indicates that a read or execute request was present, the saved PC points to the instruction that caused the pager trap. Otherwise, the saved PC has been incremented and the previous (write) instruction must be simulated by moving the saved trap data to the virtual address which caused the trap (in the right half of the trap word) before resuming.

Interpretation of TRAP Word

Bits Ø-8, trap cause are decoded as follows: Bits Ø and 1 define one of four groups each defined below:

Group Ø: TSR Ø,1 = ØØ

Bit	Meaning if ON
2	AGE = ØØX
3	AGE = Ø2X as read from
4	AGE = Ø4X C.S.T.
5	AGE = Ø6X
6	Monitor After-Loading A.R. trap

Group 1: TSR Ø,1 = Ø1

BIT	Meaning if on
3	Shared not in core
4	Page table not in core (P.T.2)
5	2nd indirect, private not in core (P.T. 3)
6	Indirect shared not in core (P.T.2 or P.T. 3)
7	Indirect page table not in core (P.T. 3)
8	Excessive Indirect pointers (>2)

Group 2: TSR Ø,1 = 1Ø

Bit	Meaning if on
2	Private Not in core
3	Write copy trap (bit 9 in P.T.)
4	User trap (bit 8 in P.T.)
5	Access trap (P.T. bit 12 = Ø or bits 1Ø-11=3)
6	Illegal Read or Execute
7	Illegal Write
8	Address Limit Register Violation or P.T. bits Ø,1=3 (illegal format)

Group 3: TSR Ø,1 = 11

	Bit	Meaning if on
	2	Private Not in core
(in	3	Write copy trap (bit 9 in P.T.)
2nd	4	User trap (bit 8 in P.T.)
or	5	Access trap (P.T. bit 12 = Ø or bits 1Ø-11=3)
3rd	6	Illegal Read or Execute
page	7	Illegal Write
table)	8	Address Limit Register Violation or P.T. bits Ø,1=3 (illegal format)

Bits 9-17 are decoded as follows:

Bit	Meaning if on
9	Parity Error
1Ø	Non-EX-MEM
11	KEY (KEY cycle in progress--console Examine, XCT, or deposit switch pushed)
12	PI (Priority Interrupt cycle in progress)
13	I (Indirect address sequence in progress)
14	R (Read request)
15	W (Write request)
16	E (Execute request)
17	EM (Exec Mode)
18	Effective Address of Request

Bits 18-35 are the virtual address which caused the trap.

Section 5 Index

SPECIAL CAPABILITIES

INTRODUCTION

There is a set of special capabilities in the TENEX system which is limited to certain processes. Each capability is separately protected and activated, and includes privileges such as the abilities to look at and/or change the running monitor, change process capabilities, and allow a fork to handle control-C interrupts from lower forks. These capabilities are assigned on a per process basis, and the status of each capability is kept in words assigned in the PSB. The number of capabilities is limited to 36, and two words are used to store the status of capabilities. For each capability there is a bit in the first word which is 1 if the capability is available to the process. In the corresponding bit in the other word, a 1 indicates that the capability is currently enabled. This allows the user to protect himself against accidental use without giving up the capability.

New forks are created upward with no special capabilities or the capabilities of the creating fork. Capabilities may be passed down from higher level forks which have the capability available. Most capabilities relate to system functions and may be passed from superior to inferior fork only if the superior itself has the capabilities. Some capabilities relate inferior to superior and so may be given to an inferior whether or not available in the superior.

Note: page number 2 is reserved for expansion of this introduction

this page to be used for future growth

FORK FREEZING

With reference to JSYS's that control whether or not a fork can execute instructions (FFORK, RFORK, HFORK, SFORK, SFRKV):

A fork may be independently frozen in two ways:

1. upon explicit request of its superior,

2. because its superior has been frozen.

We refer to these as direct and indirect freezing, respectively. When any fork is directly frozen, all of its inferiors are indirectly frozen at the same time. A fork may be both directly and indirectly frozen if it was directly frozen and its superior was subsequently also frozen. Explicitly unfreezing a fork will clear the direct freeze from it and indirect freeze from all of its inferiors unless a fork with a direct freeze is encountered. The indirect unfreeze will be cleared from that fork but not from any of its inferiors. The logical effect of this is that the explicit freezing of any fork will prevent any of its inferiors from running, and the unfreezing of such a fork will automatically resume the inferiors.

This produces the proper effect for jobs with several levels of controlling forks, for example, an EXEC running an IDDT running a test program. If the IDDT has frozen the test program (running in an inferior fork), and the user subsequently does a control-C and CONTINUE or START or REENTER), the IDDT will be resumed, but the test program will correctly remain frozen.

FFORK and RFORK may name a particular inferior fork or supply -4 as an argument to mean all immediate inferiors. That is, -4 will freeze/unfreeze all immediate inferiors, and forks below the immediate inferiors will be indirectly affected as described above. Freeze/unfreeze is never legal on any fork which is not inferior to the executing fork.

SFORK, SFRKV, and HFORK, will change the state of a fork as appropriate, but if the fork was frozen, it will remain frozen and not begin operating in the new state until resumed. For example, a frozen fork may have its PC changed by action of SFORK, but the fork will not commence running at that PC until it is unfrozen. Similarly, if HFORK is executed on a frozen fork, the fork will continue to be frozen, but when it is resumed it will then be in the halted state. The changed status is always reflected in the information returned by RFSTS, i.e. in the former example it would show the changed PC, and in the latter example, it would return the halted code in the status word.

ASSIGNED CAPABILITIES

bit	mnemonic	
		BØ-8 Job Capabilities
BØ	CTRLC	process can activate terminal key ↑C for pseudo-interrupt
B1		process can examine monitor tables with GETAB
B2		process can map the running monitor
B3	LOG	process can execute protected log functions
B4		process can map privileged pages of files
B5		process can control special devices like DP, RTI NOT IMPLEMENTED YET
		B9-17 Capabilities which can be given to an inferior whether or not the superior itself has them. Of these, B14-17 may not be changed by a process for itself.
B9		Process can do map operations on superior
B17		Unprocessed PSI's cause freeze instead of termination.
		B18-35 User Capabilities
B18	WHEEL	user is a wheel
B19	OPER	user has operator privileges
B2Ø	CONFI	user can see system confidential information (e.g. error printouts)
B21	MAINT	user can HALT TENEX This capability is sufficient for the privileged JSYS's USRIO, HSYS, LITES, and LOGOUT.

RPCAP JSYS 15Ø

Reads process capabilities words

ACCEPTS IN 1: a fork handle

RPCAP

RETURNS +1: always with special capabilities possible
in 2, special capabilities enabled in 3

Generates illegal instruction pseudo interrupt on any error conditions below.

RPCAP ERROR MNEMONICS:

FRKHX1: illegal fork handle

EPCAP JSYS 151

Enables process capabilities

ACCEPTS IN 1: a fork handle

2: special capabilities possible

3: special capabilities to enable

EPCAP

Word 2 is ignored if the fork handle points to this process. Bits Ø-8 and 18-35 of word 2 are "AND"ed with the special capabilities possible for this fork before being passed down to a lower fork. (The "AND" just reduces capabilities, does not generate errors.)

RETURNS +1: always

Generates illegal instruction pseudo-interrupt on error conditions below:

EPCAP ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: this JSYS cannot be used to manipulate a superior fork

FORKS AND SCHEDULING

INTRODUCTION

These JSYS's deal with the establishing and interrogating of the fork structure of the job.

Another JSYS related to forks (and other things) is RESET, which is described in section 2. See also PMAP, RMAP, RPACS, and SPACS in section 2 for controlling fork memory maps, and GPJFN and SPJFN for controlling primary I/O files. GTRPN (section 5) and GTRPI (section 3) provide information about the paging activities of a fork.

CFORK JSYS 152

Creates a fork inferior to this process

ACCEPTS IN 1: left half

BIT	VALUE	MEANING
Ø	1	set map as indirect to current fork
	Ø	set map to empty (all Ø)
1	1	set spec. cap. to same as current fork
	Ø	set spec. cap. to Ø
2	-	not used
3	1	set startup AC's from block of 2Ø(8) words whose address is in 2
	Ø	set startup AC's to Ø
4	1	set PC to C(AC1)(RH) and start fork
	Ø	do not start fork

right half: PC value

2: (optional) virtual address of 2Ø(8) word block

CFORK

RETURNS +1: unsuccessful, error number in 1

+2: successful, relative fork handle in 1

The new fork receives the same primary input and output JFN's as the current fork, however primary input and/or output files may be changed with SPJFN, section 2.

The bit Ø option causes the inferior to see the same address space as that of the creator. Note that the pages are not shared; changes in the inferior's page map will not be seen by the superior.

CFORK ERROR MNEMONICS:

FRKHX6:	all relative fork handles used up
CFRKX2:	illegal to start a fork unless bit Ø is set with CFORK
CFRKX3:	no room in system

KFORK JSYS 153

Kills one or more Forks

ACCEPTS IN 1: a fork handle

KFORK

Generates an illegal instruction pseudo-interrupt on error conditions below.
Killing a fork releases all truly private memory acquired by the fork and its PSB; it also passes back any terminal key assignments which were acquired from another fork and releases JFN's fork has created.

KFORK ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork
KFRKX1:	this JSYS cannot kill the top level fork
KFRKX2:	suicide not legal

SPLFK JSYS 314

SPLice ForK structure.

Accepts in 1: fork handle of new superior

2: fork handle of fork to become inferior

SPLFK

Returns +1: unsuccessful, error number in 1
+2: successful, a handle in 1

If successful, a handle is returned which may be used by the new superior fork (AC1) to refer to its new inferior (AC2).

Restrictions: The "new superior" must be either the fork performing the SPLFK or an inferior of it. The "new inferior" must be strictly an inferior of the executing fork. The "new superior" must not be the same fork as the "new inferior", nor may the "new superior" be an inferior of the "new inferior". The new inferior and all its inferior forks will be frozen after the SPLFK.

SPLFK ERROR MNEMONICS:

SPLFK1:	new superior not self or inferior
SPLFK2:	new inferior not already inferior
SPLFK3:	new inferior not inferior to new superior
FRKHX1:	illegal fork handle

RFORK JSYS 155

Resumes one or more forks which had been frozen (by melting them)

ACCEPTS IN 1: a fork handle

RFORK

RETURNS +1: always

This does nothing if the referenced fork(s) were not frozen.

Generates an illegal instruction pseudo-interrupt on error conditions below.

RFORK ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork

RFSTS JSYS 156

Reads fork status

ACCEPTS IN 1: a fork handle

RFSTS

RETURNS +1: always; status word in 1, PC in 2

If the handle is in legal form but unassigned in the current process, RFSTS returns normally with -1 in the LH of 1.

Generates illegal instruction PSI on error conditions below

RFSTS ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX3:	fork handle cannot reference multiple forks for this JSYS

FORK STATUS WORD

LEFT HALF

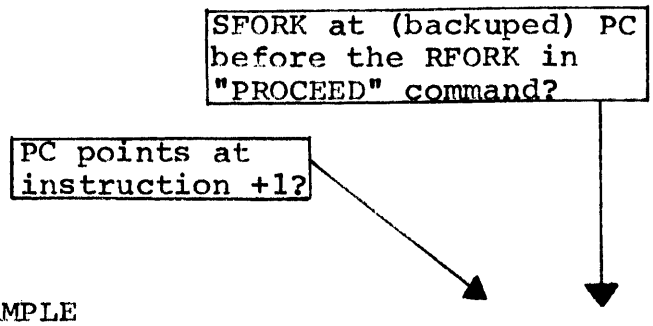
(sign bit on means frozen, off melted)

0 runable
1 dismissed for I/O,
2 dismissed by voluntary fork termination (HFORK or HALTF) or never started
3 dismissed by forced fork termination
4 dismissed waiting for another fork to terminate
5 dismissed for a specific amount of time (* currently returns status 1)
6 breakpoint (* currently returns status 2)

RIGHT HALF

The pseudo interrupt channel # which caused the forced fork termination

Fork States



CODE	STATUS	PC-5 USERMODE	EXAMPLE		
Ø	Running	Ø	Interrupt out of long JSYS like RESET,JFNS	YES	NO
		1	Normal Interrupt	NO	NO
1	IO WAIT	Ø	Interrupt out of PBIN, GTJFN,JOUT	YES	NO
		1	----------		
2	Voluntary Termination	Ø	HFORK, HALTF, (BPT)*	YES	YES
		1	----------		
3	Involuntary Termination	Ø	Illegal Instruction JRST 4,	YES If write	YES
		1	Illegal Memory Reference	trap	YES
4	FORK WAIT	Ø	Interrupt out of WFORK	YES	NO
		1	----------		
5**	Dismissed For ΔT	Ø	Interrupt out of DISMS	YES	NO
		1	----------		
6*	At a Breakpoint	Ø	BPT	YES	YES
		1	----------		

* BPT currently reports a status code of 2 and must be handled as a special case.

** DISMS currently reports a status of 1.

Plans exist to implement codes 5 and 6.

SFORK JSYS 157

Starts a fork

ACCEPTS IN 1: a fork handle

2: virtual address (for starting PC)

SFORK

RETURNS +1: always, if fork was frozen in any manner, SFORK simply changes the fork PC; the fork must be resumed by RFORK

Generates illegal instruction pseudo-interrupt on error conditions below.

SFORK ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: this JSYS cannot be used to manipulate a superior fork
FRKHX3: fork handle cannot reference multiple forks for this JSYS
FRKHX4: fork already running

SFACS JSYS 16Ø

Sets Fork AC'S

ACCEPTS IN 1: a fork handle

2: pointer to a 2Ø(8) word table in address space of this process containing values to be set into AC's of specified fork

SFACS

RETURNS +1: always

Specified fork must not be running.
Generates illegal instruction pseudo interrupt on error conditions below.

SFACS ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork
FRKHX3:	fork handle cannot reference multiple forks for this JSYS
FRKHX4:	fork already running

RFACS JSYS 161

Read Fork AC'S

ACCEPTS IN 1: a fork handle

2: a pointer to a 20(8) word table in address space of this process where the AC values of the specified fork will be stored.

RFACS

RETURNS +1: always

Generates illegal instruction pseudo-interrupt on error conditions below.

RFACS ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork
FRKHX3:	fork handle cannot reference multiple forks for this JSYS
FRKHX4:	fork already running

HFORK JSYS 162

Halts one or more forks

ACCEPTS IN 1: a fork handle

HFORK

RETURNS +1: always, just sets PC to this value if HFORK halted this process

Sets status word(s) for addressed fork(s) bits 1-17 to 2.

Note: See HALTF to halt current fork with no arguments (and thus no clobbered AC's).

Generates an illegal instruction pseudo-interrupt on error conditions below.

HFORK ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork
FRKHX5:	addressed fork(s) was (were) already halted

WFORK JSYS 163

Waits for one or more forks to terminate (voluntarily or involuntarily)

ACCEPTS IN 1: a fork handle

WFORK

RETURNS +1: when one or more of the addressed forks terminate

WFORK for multiple fork handles NOT IMPLEMENTED YET. Generates an illegal instruction pseudo-interrupt on error conditions below.

WFORK ERROR MNEMONICS

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork

GFRKH JSYS 164

Gets a fork handle useable in this process for a fork currently not known to this process

ACCEPTS IN 1: a fork handle of some fork which does have a handle (call it H) on the desired fork

2: the handle H, must be 400001-400030 and must point to an existent fork

GFRKH

RETURNS +1: unsuccessful, error number in 1
+2: successful, a useable handle H' in 1

GFRKH ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	this JSYS cannot be used to manipulate a superior fork
FRKHX3:	fork handle cannot reference multiple forks for this jsys
GFRKX1:	the handle H was illegal
FRKHX6:	all relative fork handles used up

RFRKH JSYS 165

Releases a fork handle or all fork handles in this process

ACCEPTS IN 1: a fork handle

RFRKH

RETURNS +1: always

RFRKH ERROR MNEMONICS:

FRKHX1:	the handle in 1 was illegal

GFRKS JSYS 166

Gets fork structure from a given starting point downward
NOT IMPLEMENTED YET

ACCEPTS IN 1: the fork handle of the starting point

in 2: LH:
B0 on - get relative fork handles for each fork in structure

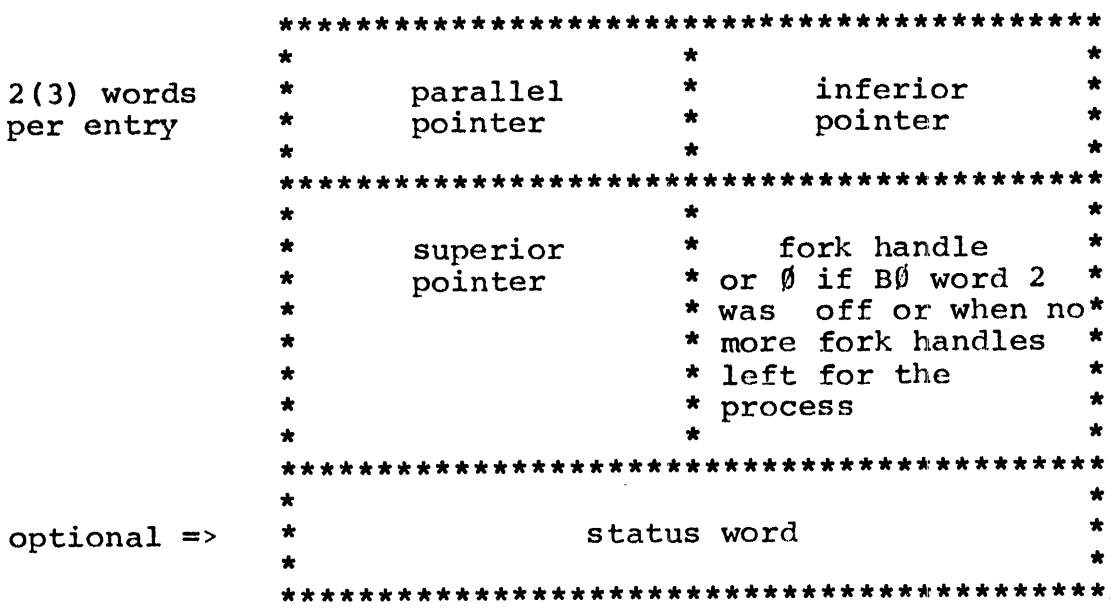
B1 on - get status for each fork in structure

RH:
virtual address of start of table in address space of this process where the structure will get stored

GFRKS

RETURNS +1: always

Table format



Note: "pointers" in above table are core addresses of other table entries, or Ø if no such structure.

Generates illegal instruction pseudo-interrupt on error conditions below.

GFRKS ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX3:	fork handle cannot reference multiple forks for this JSYS

DISMS JSYS 167

Dismiss this process for at least the specified amount of time.

ACCEPTS IN 1: a number of milliseconds for which the process is to be dismissed

DISMS

RETURNS +1: when the elapsed time is up

The time resolution is obviously limited to the clock in use on the PDP-10.

HALTF JSYS 170

Halts this process

HALTF

Sets PC to caller+1 and halts process. Sets status word for this process bits 1-17 to 2.

If an attempt to execute a HALTF at the top level is made without WHEEL or OPER special capability enabled, the top level process is replaced by the TENEX EXEC which is started at its initialization entry.

BPT JSYS 304

Breakpoint

Implementation incomplete. Current behavior is identical to HALTF.

WAIT JSYS 3Ø6

Dismiss this process indefinitely (i.e., wait) - does not return. However, if the pseudo-interrupt system is enabled, the process can be interrupted out of the WAIT. On DEBRK, the process continues WAITing for the next interrupt.

Section 6 Index

SAVE AND ENVIRONMENT DUMP FILES

INTRODUCTION

TENEX handles SAVE files of three forms: 1Ø/5Ø SAVE format, regular (non-sharable) TENEX SAVE files, and TENEX sharable SAVE files. The latter can exist on DSK: only.

TENEX SAVE FORMAT	1Ø/5Ø SAVE FORMAT
IOWD length, location data	IOWD length, location data
IOWD length, location data	
E: XWD length of entry vector, pointer to first entry (possible MISC other words from PSB)	E: JRST start - start JRST reenter - reenter
The entry vector is in the address space of the SAVE file	For a 1Ø/5Ø format SAVE file, entry vector relative position Ø => use JOBSA (12Ø) 1 => use JOBREN (124) all other relative positions illegal

Note: Files actually SAVed by a 1Ø/5Ø monitor have the contents of location 41 shifted to location 122. After GETing such a file on TENEX, it may be necessary for the user to move this word to location 41.

TENEX Shared File Format

SSAVE creates files which contain sequences of pages, whereas SAVE stores words. An SSAVE file has two parts: the actual pages containing data (pages 2,3 ... of the file), and pages Ø and 1 which contain length, mapping information and the entry vector for use when a GET is done. Usually all of the control information can be held in just page Ø of the file. Thus, most SSAVE'd files will not have a page 1.

Page Ø (and 1 if needed):

```
     Word Ø: 1ØØØ,,N                    N is the number of data pages

     Word 1: )
     Word 2: )
     Word 3: )
          .  )--- Mapping information
          .  )
          .  )
          .  )
     Word N: )

     Word N+1: LENGTH,,ADDRESS          TENEX entry vector

Pages 2, 3, 4, ...                      Data Pages
```

Each mapping information word has three fields which are used in the GET JSYS to set up a PMAP:

```
     bits Ø-8:      Access bits, left 9 bits of AC3 in PMAP
     bits 9-17:     Fork page, AC2 of PMAP
     bits 18-35:    File page, AC1  of PMAP
```

GET JSYS 200

Gets a SAVE file, copying or mapping it into the fork as appropriate. Works with TENEX and 10/50 SAVE formats and TENEX sharable SAVE format. Updates fork's entry vector from file.

ACCEPTS IN 1: LH: fork handle
RH: JFN

GET

RETURNS +1: always

When a GET is performed on an SSAVE file, pages from the file are mapped into the fork, and the previous contents of the map are removed. GET does not disturb existing pages for which the file contains no data, but if the file contains data for part of a page only, the rest of the page is zeroed.

When a GET is performed on a SAVE file, individual words are written into the fork, however, SAVE files usually do not contain 0 words. Thus, GETing a SAVE file will never clear memory.

GET never loads the accumulators.

Generates an illegal instruction PSI on error conditions below. Can cause several PSIs or fork termination on certain file conditions.

GET ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks
GETX1:	illegal SAVE format
GETX2:	no room in Special Pages Table, can only occur with a TENEX sharable SAVE file.
SSAVX1:	SSAVE file not from DSK:
	All file errors can occur.

SFRKV JSYS 201

Starts fork using entry vector

ACCEPTS IN 1: a fork handle

2: a relative position in entry vector to use for start address (Ø-n). Ø is always primary start address, 1 is reenter address.

SFRKV

RETURNS +1: always

The fork is started at the indicated position IN the entry vector, NOT at the location of which the entry vector word points to.

If the process has 1Ø/5Ø format entoy vector (JRST in LH), for example because it was GETed from a 1Ø/5Ø format SAVE file, entry vector relative position Ø means use contents of JOBSA=12Ø as start address, and 1 means use contents of JOBREN=124 as reenter address.

Generates illegal instruction PSI on error conditions below.

SFRKV ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate a superior fork
FRKHX3:	cannot be used to manipulate multiple forks
FRKHX4:	fork already running
SFRVX1:	illegal relative position

SAVE JSYS 2Ø2

SAVEs (non-sharably) areas of a process. Notes whether process is TENEX or 1Ø/5Ø format by its entry vector word (JRST in LH for 1Ø/5Ø format) and appropriately formats last word(s) of SAVE file.

ACCEPTS IN 1: LH: fork handle
RH: JFN

2: a single table word (see below), or LH=Ø and RH pointing to table. In the latter case, the table is terminated with a Ø word.

SAVE

RETURNS +1: always

Table Word Format

LH: length of block to SAVE
RH: location of first word in block

Nothing is SAVEd for all zero areas or for non-existent pages. It is thus possible to SAVE all assigned non-Ø core memory with the single table word 777760,,20. Note that it is generally not necessary to SAVE zero core because the monitor creates a zero page on the first reference to a non-existent page.

Generates an illegal instruction PSI on error conditions below. Can also general several PSI's or fork termination on certain file conditions.

CAUTION: SAVE will not save AC's, JFN is closed and released.

SAVE ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate a superior fork
FRKHX3: cannot be used to manipulate multiple forks

All file errors can occur.

SSAVE JSYS 2Ø3

Sharable SAVE, for creating shared programs, e.g., subsystems.

ACCEPTS IN 1: LH: fork handle
RH: JFN

2: a single table word (see below), or LH=Ø and RH pointing to table. In latter case table is terminated with a Ø word.

3: flags (none specified yet, 3 should be Ø)

SSAVE

RETURNS +1: always

Table Word Format

LH: negative of number of pages in group
B18-26: protection to be used for these pages on GET
B18: set copy on write
B19: limit access per current access of pages
B2Ø-22: read, write, xct.

Examples: 52Ø for shared page with copy on write
12Ø for share page, write protected

B27-35: first page number in group

Creates a TENEX sharable SAVE format file on the given JFN, copying (not sharing) pages from the given fork.

The inverse of SSAVE is GET.

SSAVEing unassigned pages saves nothing for those pages but does not produce an error. SSAVE will not save AC's.

Generates illegal instruction PSI on error conditions below. Can also cause several PSI's or fork termination on certain file conditions.

CAUTION: Closes and releases JFN for SAVE files but not for SSAVEd files.

SSAVE ERROR MNEMONICS:

FRKHX1: illegal fork handle
FRKHX2: cannot be used to manipulate superior fork
FRKHX3: cannot be used to manipulate multiple forks
SSAVX1: file not on DSK:
SSAVX2: illegal page count in table (>512)

All file errors can occur.

Entry Vector

An entry vector word can be either:

LH: length of vector (1-777)
RH: location of first word of vector

or (10/50 format)

LH: JRST (254000)
RH: ignored by TENEX

Each fork (process) has an entry vector word stored in its PSB. The word is 0 if it has never been set.

SEVEC JSYS 204

Sets Entry VECtor

ACCEPTS IN 1: fork handle

2: a legal entry vector or 0

SEVEC

RETURNS +1: always

Generates illegal instruction PSI on error conditions below.

SEVEC ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate superior fork
FRKHX3:	cannot be used to manipulate multiple forks
SEVEX1:	unreasonable length entry vector

SCVEC JSYS 3Ø1

Set Compatibility entry VECtor and UUO locations

ACCEPTS IN 1: a fork handle

2: XWD entry vector length, entry vector location

3: XWD UUO location, PC location

SCVEC

RETURN +1: always

If the location specified in 2 is non-Ø, the monitor will transfer to that location on any monitor UUO (opcode Ø4Ø-Ø77) after moving the contents of 4Ø and the return PC to the locations specified by the LH and RH of 3 respectively. The entry vector length is retained but not used by the monitor. If the length and location of the entry vector are set to Ø, the next monitor UUO will cause the compatiblity package (<SUBSYS>PA1Ø5Ø.SAV) to be merged and the locations to be set from the file. At that time, the UUO and PC locations are set from the RH of words 3 and 4 respectively of the PA1Ø5Ø file entry vector. That is, the PA1Ø5Ø entry vector is structured as follows:

word Ø - entry address for interpreting UUO'S

word 1 - initial entry for setup and first UUO

word 2 - entry for GET SHR file routine

word 3 - location to receive C(4Ø) on UUO

word 4 - location to receive return PC on UUO

word 5 - entry for MAKE SHR file routine.

words 6 and 7 - communication for handling ↑C, START sequences, by cooperation between PA1Ø5Ø and EXEC.

All fork handle errors may occur.

GEVEC JSYS 2Ø5

Gets Entry VECtor

ACCEPTS IN 1: fork handle

GEVEC

RETURNS +1: with specified fork's entry vector word in 2

Generates illegal instruction PSI on error conditions below.

GEVEC ERROR MNEMONICS:

FRKHX1:	illegal fork handle
FRKHX2:	cannot be used to manipulate superior fork
FRKHX3:	cannot be used to manipulate multiple forks

GCVEC JSYS 3ØØ

Get Compatibility entry VECtor and UUO locations.

ACCEPTS IN 1: a fork handle

GCVEC

RETURNS +1: always, with
2: LH: entry vector length
RH: entry vector location

3: LH: UUO location
RH: PC location

All fork handle errors may occur.

Section 7 Index

INPUT/OUTPUT CONVERSION JSYS'S

INTRODUCTION

This section of the manual defines those JSYS's which do input/output conversion. JSYS's are available to convert in both directions between ASCII text (in core or on a file) and integer numbers, floating point numbers, and TENEX internal dates and times.

NOUT JSYS 224

Number output (integer)

ACCEPTS IN 1 : a TENEX destination designator

2: a number

3: left half:
B0 : print magnitude if set
B1: print plus sign on positive number (or magnitude) if set
B2: leading filler if set, trailing filler if clear
B3: leading 0's if set, leading blanks if clear
B4: do output per B5 on column overflow (and take error return) if set, no output on column overflow if clear
B5: asterisks on column overflow if set, all necessary digits on column overflow if clear
B11-B17: number of columns including sign (if 0 means print as many columns as necessary)
right half:
radix (2-16)

NOUT

RETURNS +1: on error, error # in 3

+2: successful, updated string pointer in 1 (if pertinent)

NOUT ERROR MNEMONICS:

NOUTX1: illegal radix
NOUTX2: column overflow
DESX1: illegal value for designator
DESX2: tty not available to this job
DESX3: no name for this JFN
DESX5 : not open

NIN JSYS 225

Input number (integer). Leading spaces ignored, sign handled correctly, terminates on first non-numeric character after numeric characters.

ACCEPTS IN 1: a TENEX source designator

3: a radix (2-1Ø)

NIN

RETURNS +1 : unsuccessful, error # in 3
updated string pointer (if pertinent) in 1

+2: successful, number in 2, updated string pointer (if pertinent) in 1

NIN ERROR MNEMONICS:

IFIXX1:	illegal radix
IFIXX2:	first character not a digit in specified radix, not a number
IFIXX3:	overflow (low 36 bits returned in 2)
DESX1:	illegal value for designator
DESX2:	tty not available to this job
DESX3:	no name for this JFN
DESX5:	not open

Floating Point Conversion JSYS's

JSYS's for Floating Point I/O Conversion are available for both single precision and double precision numbers. The double precision binary representation is a superset of the DEC double precision representation to permit a wider exponent range. The normal DEC double precision format permits numbers with an exponent range of approximately 1Ø↑+38 to 1Ø↑-32. The extended format permits exponent ranges of approximately 1Ø↑+/-986Ø. However, internal limitations in double precision floating input make it useful only for the range 1Ø↑+/-99. While double precision floating output will work for the full range, it gets increasingly slower and less accurate beyond 1Ø↑+/-99. The accuracy of the double precision routines is generally 14 decimal digits. The single precision output conversion routine is accurate to +/-1 in the eighth digit and exact to seven digits. The single precision input conversion routine is accurate to eight digits.

Extended Range Format

Numbers in the exponent range 1Ø↑+38 to 1Ø↑-32 are represented in standard DEC double precision format. Numbers in extended range are flagged by the sign bit of the second word being set. (This bit is normally cleared in the standard DEC double precision format and is both unaffected by and ignored in all standard DEC double precision arithmetic operations.) With extended range exponents, the sign and high order bits of the exponent are stored in bits 1-8 of the second word. Bit 1 is the sign bit, off for positive and on for negative exponents. For positive double precision numbers, the full exponent may be obtained by concatenating bits 1-8 of the second word with bits 1-8 of the first word and then subtracting 2ØØ(8). (The representation in extended format is still excess 2ØØ(8).) For negative double precision numbers, the low order exponent bits in 1-8 of the first word are negated but the high order exponent bits in the second word are to be interpreted literally. This permits the PDP-1Ø instruction DFN (double floating negate) to do exactly the right thing in negating an extended format number.

FLIN JSYS 232

Floating Point Number Input

Ignores leading spaces, terminates on first character which can't be part of a floating point number.

ACCEPTS IN 1: TENEX source designator

FLIN

RETURNS +1: unsuccessful, error # in 3, updated string pointer in 1 (if pertinent)

+2: successful, single precision floating point number in 2, updated string pointer in 1 (if pertinent)

FLIN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	tty not available to this job
DESX3:	no name for this JFN
DESX5:	not open
FLINX1:	first character not formatting, digit,sign, or "."
FLINX2:	number too small, floating underflow, returns Ø
FLINX3:	number too big, floating overflow, returns largest number, correct sign
FLINX4:	"." only input or "-.Q", ill-formed number

DFIN JSYS 234

Floating Point Number Input, Double Precision

ACCEPTS IN 1: TENEX source designator

DFIN

RETURNS +1: unsuccessful, error # in 4, updated string pointer in 1 (if pertinent)

+2: successful, double precision floating point number (extended format where necessary) in 2 and 3, updated string point in 1 (if pertinent)

DFIN ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	tty not available to this job
DESX3:	no name for this JFN
DESX5:	not open
FLINX1:	First character not formatting, digit,sign, or "."
FLINX2:	number too small, floating underflow, returns Ø
FLINX3:	number too big, floating overflow, returns largest number, correct sign
FLINX4:	"." only input or "-.Q", illformed number

FLOUT JSYS 233

Floating Point Number Output

ACCEPTS IN 1: TENEX destination designator

2: normalized, single precision floating point number

3: format control word (see below)

FLOUT

RETURNS +1: unsuccesful, error # in 3, updated string pointer in 1 (if pertinent).

+2: successful, updated string pointer in 1 (if pertinent).

FLOUT ERROR MNEMONICS:

DESX1:	illegal value for designator
DESX2:	tty not available to this job
DESX3:	no name for this JFN
DESX4:	not open
FLOTX1:	column overflow field 1 or 2
FLOTX2:	column overflow field 3
FLOTX3:	illegal format specif. e.g., exponent requested but field 3 zero.

DFOUT JSYS 235

Floating Point Number Output, Double Precision

ACCEPTS IN 1: TENEX destination designator

2: a normalized double precision

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

4: format control word (see below)

DFOUT

RETURNS +1: unsuccessful, error # in 4, updated string pointer in 1 (if pertinent).

+2: successful, updated string pointer in 1 (if pertinent).

DFOUT ERROR MNEMONICS:

DESX1: illegal value for designator
DESX2: tty not available to this job
DESX3: no name for this JFN
DESX5: not open
FLOTX1: column overflow field 1 or 2
FLOTX2: column overflow field 3
FLOTX3: illegal format spec, e.g. exponent requested but field 3 zero.

Floating Output Format Control Word

Free Format

The most common format control used with FLOUT and DFOUT is free format. This is specified by setting bits 18-23 of the format control word to Ø. In the standard case, the entire format control word is set to Ø; however, most of the other fields may be specified to force a particular setting. The effect of this is that all numbers of magnitude <1Ø↑6 but >=1Ø↑-4 are printed in a typical Fortran F format (with some exceptions). If the number is an exact integer, the number is printed with no terminating decimal point unless point (B6) is specifically requested. If the number is a fraction in the range above it is printed as .xxxx with no Ø preceding the ".". Non-significant trailing zeros in the fraction are never printed. A maximum of 7 digits total will be printed if field 2 is not specified. The sign of the number is printed only if negative. The number zero is printed as "Ø".

If the number is beyond the range indicated above, the number is printed in a typical Fortran E format (with some exceptions). The exponent is printed as Esxx where xx are the digits of the exponent and s is the sign which is printed only for negative exponents. The same exceptions about when to print the "." and suppression of trailing, non-significant zeros as above apply.

Another form of free format similar to that above is invoked by having a non-zero field B13-B17 of the format control word. The number in this field specifies essentially at what place rounding should occur. If this number is 7, the behavior is exactly the same as if it were Ø as above. If this number is <7, rounding will occur at the specified place but the printout format will be as above with a maximum of 7 digits printed. (e.g. 1236Ø with a rounding specification of 3 will print as 124ØØ). If this number is >7, rounding will again occur at the specified position but more than 7 digits maximum will be printed. In fact, digits will be printed until either the rounding specification number is reached or trailing non-significant zeros are reached.

General Format Control

This section describes the various controls available for floating point output format when free format is not desired.

B18-B23 The number of characters in field 1 (normally before the decimal point, see field 1 justification control below).

B24-B29 The number of digits after the decimal point, which is field 2

B3Ø-B35 The number of characters in the exponent field, which is field 3

B13-B17 Specifies digit position for rounding, Ø=>12, 37=>no limit

BØ-B1 field 1 sign control, first character position is always used for "-" on negative numbers.
on positive numbers:
ØØ - first character position is digit position
1Ø - first character position is "+"
Ø1 or 11 - first character position is space

B2-B3 field 1 justification control
ØØ - fill field to left with spaces
Ø1 - fill field to left with Ø's
1Ø - fill field to left with *'s
11 - left justify whole number up to "." by filling after field 3 with spaces, i.e. field 1 wraps around to after field 3

B4 If set print at least one digit (Ø if necessary) in field 1

B5 Prefix the number with dollar sign ($)

B6 If Ø print no decimal point,
If 1 print decimal point

B7-B8 exponent field control
ØØ - no exponent field
Ø1 - print "E" as first character of exp field
1Ø - print "D" as first character of exp field
11 - print "*1Ø↑" as first chars of exp field

B9-B1Ø exponent sign control
first character position of exp field is always used for "-" on negative exponents.
on position exponents:
ØØ or 11 - first character position after prefix is digit position
Ø1 - first character position after prefix is "+"
1Ø - first char position after prefix is space

B11 If Ø do no additional output on column overflow;
If 1 go to free format on overflow of field 1, expanded exponent format on field 3 overflow

B12 (not currently used)

Date and Time Conversion JSYSs

As described in the introduction to this manual, internal date and time are at the Greenwich meridian. The date is days since November 17,1858; the time is seconds since midnight, with midnight=Ø. The date changes at the transition from 11:59:59 PM to 12:ØØ:ØØ midnight. Each is an 18-bit quantity, with the date in the LH and the time in the RH of a JSYS argument word.

When converting 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 last Sunday in April to 3:59:59AM on the last Sunday in October.

The first two JSYS's in this group, IDTIM and ODTIM, convert date and time between text strings (in core or on a file) and internal format. These should satisfy most, if not all, users. However, there are four more JSYS's which are subsets of IDTIM and ODTIM. These 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.

Although the typical user can ignore them, time zones occur in the calling sequences of the latter four JSYS's. The internal representation of a time zone is as a number between -12 and 12 decimal, representing a number of hours west of Greenwich. For example, EST is zone 5. Zones -12 and 12 represent the same time difference but on opposite sides of the international date line.

ODTIM JSYS 22Ø

Output date and time. Converts internal date and/or time to text.

ACCEPTS IN 1: TENEX destination designator

2: internal date and time,
or -1 to use current date and time

3: format flags, described below

ODTIM

RETURNS +1: always, with updated string pointer in 1 if pertinent

Generates illegal instruction PSI on error conditions below.

ODTIM ERROR MNEMONICS:

DATEX6: system has no date and time (and -1 was given)
TIMEX1: illegal time (greater than 24 hours)

All I/O errors are also possible and produce PSI's or fork termination as described for BOUT in section 2.

ODTIM format option flags (AC 3):

B0 on, omit date from output and ignore B1-B8
B1 on, output day of week
B2 on, use full text for weekday
   off, use 3-letter abbreviation
B3 on, output month as number and ignore B4
B4 on, use full text for month
   off, use 3-letter abbreviation for month
B5 on, output 4-digit year
   off, output 2-digit year if between 1900 and 1999
B6 on, output day of month after month
   off, output day of month before month
B7 and B8: date punctuation control:
   both off, use dashes, as 13-APR-70
   B7 on, use spaces, as 13 APR 70
        a comma is also used if B6 is on, as APRIL 13, 1970
   B8 on, use slashes, as 5/13/70
B9 on, omit time from output and ignore B10-B13
B10 on, omit seconds
    off, include seconds, preceded by a colon
B11 on, use 12-hour time and AM-PM
    off, use 24-hour time
B12 on, no colon between hours and minutes
B13 on, and "-" and time zone to time, e.g. "-EDT" or "-PST"
B17 off, columnate: use leading spaces and zeros so output is constant width regardless of the particular date or time, for use in printing tables, etc. (Note: full month and weekday texts are not columnated)
    on, suppress columnation: omit leading spaces and zeros, producing output appropriate for inclusion in an isolated message

Zero in 3 produces a columnated printout of the form " 3-APR-70 1:06:03", for example.

For convenience, -1 in 3 is taken as 336021000000, which produces,
for example:
"FRIDAY, FEBRUARY 2, 1973 1:06:03" EST.

More examples:

contents of 3	typical text
202201000000	"FRI 3 APR 70 1:06"
336321000000	"FRIDAY, APRIL 3, 1970 1:06AM-EST"
041041000000	"3/4/70 106:03"
041040000000	" 3/04/70 106:03"

IDTIM JSYS 221

Input date and time. Converts text to internal date and time

ACCEPTS IN 1: TENEX source designator

2: format option flags, described in detail below, Ø for normal case

IDTIM

RETURNS +1: unsuccessful, error number in 2, updated string pointer in 1 if pertinent

+2: success
1: updated string pointer if pertinent
2: internal format date and time

IDTIM does not permit inputting either the date or the time separately, nor does it permit conversions for time zones other than the local one (except if the time zone is specified in the input string). See IDTNC and IDCNV, below, for these functions.

IDTIM ERROR MNEMONICS:

DILFX1:	illegal date format
TILFX1:	illegal time format
DATEX1:	illegal year (out of range 1858 to 2576)
DATEX3:	day of month too large
DATEX5:	illegal date (out of range 17 NOV 1858 to 7 AUG 2576 GMT)

All I/O errors are also possible and produce PSI's or fork termination as described under BIN in section 2.

IDTIM Input Format. If all of the format option flags are off, IDTIM accepts most any reasonable date and time syntax. the following are examples of valid dates:

```
17-APR-7Ø
APR-17-7Ø
APR 17 7Ø
APRIL 17, 197Ø
17 APRIL 7Ø
17/5/197Ø
5/17/7Ø
```

Times:

```
1:12:13
1234
16:3Ø              (4:3ØPM)
163Ø
1234:56
1:56AM
1:56-EST
12ØØNOON
12:ØØ:ØØAM         (midnight)
11:59:59AM-EST     (late morning)
12:ØØ:Ø1AM         (early morning)
```

"AM" or "PM" may follow a time not greater than 12:59:59, "NOON" or "MIDNIGHT" may follow 12:ØØ:ØØ, and any time may be followed by a dash and a time zone (EST, EDT, CST, CDT, MST, MDT, PST, PDT, GMT (Greenwich), GDT, DAYLIGHT (local daylight saving), STD or STANDARD (local standard)).

All strings (months, time zones, AM-PM-NOON-MIDNIGHT) may be represented by any non-ambiguous abbreviation, e.g. D=DECEMBER, M=MIDNIGHT.

Spaces are ignored before or between fields whenever they could not terminate the input string; they are thus not allowed before colons, AM-PM-NOON-MIDNIGHT, the dash before the time zone, or the time zone. A tab is also allowed between the date and time.

The input string may be terminated by any non-alphanumeric character.

The format option flags impose various restrictions on the above, and determine the interpretation of ambiguous cases.

IDTIM Format Option Flags (AC 2):

B1 on, disallow numeric month and ignore B2-B3
B2 off, consider the first number to be the month
in a date such as "1/2/7Ø"
on, consider the second number to be the month
B3 off, if an invalid date can be successfully interpreted
by assuming the day and month to be in the opposite
order than that specified by B2, do so.
This applies, for example, to "31/1/7Ø" with B2 off
on, give error return for such dates
B7 and B8 govern inclusion of seconds in time:
both off, optional
B7 on, illegal
B8 on, mandatory
Note: seconds must always be preceded by a colon
B9 and B1Ø govern use of colon between hours and minutes:
both off, optional
B9 on, illegal
B1Ø on, mandatory
B11 and B12: when B7 thru B1Ø are all off, times containing a single colon are ambiguous. These flags determine whether the colon is considered to separate the hours and the minutes or the minutes and the seconds in such cases
Both off, take as HH:MM if first field is small enough, else as HHMM:SS
B11 on, always take as HHMM:SS
B12 on, always take as HH:MM (giving error return if first field is too big). This differs from B7 on in that seconds can be included if preceded by a second colon
B14 on, disallow 24-hour time and make AM-PM mandatory
B15 on, disallow AM-PM-NOON-MIDNIGHT
B16 on, disallow time zone

ODTNC JSYS 23Ø

Output date and/or time without conversion from internal. A subset of ODTIM, permitting printing of dates and times not stored in internal format, and giving the caller control over the time zone printed

ACCEPTS IN 1: TENEX destination designator

2: LH: real year
RH: month (January=Ø)

3: LH: day of month (first day=Ø)
RH: day of week (Monday=Ø)
if output thereof requested

4: LH: if time zone printout requested:
B1: on for daylight savings
B12-B17: time zone
RH: local time (seconds since midnight)

5: format option flags, same as ODTIM (above).
Note: the only time zones which can be printed (flag B13) are Greenwich and those in the USA

ODTNC

RETURNS +1: always, with updated string pointer in 1 if pertinent.

Generates illegal instruction PSI on the error conditions below.

ODTNC ERROR MNEMONICS:

DATEX1:	illegal year
DATEX2:	month too large
DATEX3:	day of month too large
DATEX4:	day of week too large
ZONEX1:	illegal time zone (not between -12 and 12)
ODTNX1:	unprintable time zone (not in USA or Greenwich)

All I/O errors can occur and produce PSI's or fork termination as described for BOUT in section 2.

IDTNC JSYS 231

Inputs date and time without conversion to internal. Permits input of date or time separately, which is not possible with IDTIM because neither can be converted to internal without the other

ACCEPTS IN 1: TENEX source designator

2: format option flags:
those described under IDTIM apply, and also:
BØ on, do not input date, ignore B1-B3
B6 on, do not input time, ignore B8-B16

IDTNC

RETURNS +1: unsuccessful, error code in 2, updated string pointer in 1 (if pertinent).

+2: success, updated string pointer in 1 (if pertinent).
if date input not suppressed:
2: LH: real year
RH: month (January=Ø)
3: LH: day of month (first day=Ø)
RH: day of week (Monday=Ø)
If time input was not suppressed:
4: RH: time as seconds since midnight
BØ: off unless a time zone was input
B1: on if a daylight saving time zone (e.g. EDT) was input
B2: off unless a time zone was input (redundant bits BØ and B2 are for compatability with ODCNV)
B12-B17: time zone, if one was input, else local time zone

IDTNC ERROR MNEMONICS:

DILFX1: illegal date format
TILFX1: illegal time format

IDTNC does not detect certain errors in date input, such as day 31 of a 3Ø-day month. These errors are detected in IDCNV.

IDTNC can cause all I/O errors, which cause PSI's or fork termination as described in section 2 for BIN.

ODCNV JSYS 222

Output date and time conversion. Breaks internal date and time into separate numbers for local weekday, day, month, year, and time. Does not convert to text. Gives caller the option of explicitly specifying the time zone and whether or not to apply daylight saving

ACCEPTS IN 2: internal date and time,
or -1 to use current date and time

4: (Ø for normal case)
BØ: off, use daylight savings or not
as appropriate for given date
on, see B1
B1: on to use daylight savings if BØ is on
B2: off, use local time zone
on, see B12-B17
B12-B17: time zone to use if B2 on

ODCNV

RETURNS +1: always
2: LH: real year
RH: month (Ø=January)
3: LH: day of month (Ø=first day)
RH: day of week (Ø=Monday)
4: BØ and B2: ON for compatibility with IDCNV
B1: on if daylight savings was applied
B12-B17: time zone used
RH: local time in seconds since midnight

Generates illegal instruction PSI on the error conditions listed below.

ODCNV ERROR MNEMONICS:

DATEX6: system has no date and time (and -1 was given)
TIMEX1: illegal time (greater than 24 hours)
ZONEX1: illegal time zone (not between -12 and 12)

IDCNV JSYS 223

Input date and time conversion. Translates separate numbers for local year, month, day, and time into internal format (Greenwich) date and time

ACCEPTS IN 2: LH: real year
RH: month (January=Ø)
3: LH: day of month (first day=Ø)
4: LH (Ø for normal case):
BØ: off, use daylight saving if appropriate
on, see B1
B1: on to use daylight saving if Ø on
B2: off, use local time zone
on, use time zone in B12-B17
B12-B17: time zone if B2 on
RH: local time in seconds since midnight

IDCNV

RETURNS +1: unsuccessful, error number in 1

+2: success
2: internal date and time
4: BØ and B2: ON for compatibility with ODCNV
B1: on if daylight savings was applied
B12-B17: time zone used

IDCNV ERROR MNEMONICS:

DATEX1:	illegal year (out of range 1858 to 2576)
DATEX2:	month too large
DATEX3:	day of month too large
DATEX5:	illegal date (out of range 17-NOV-1858 to 7-AUG-2576 GMT)
TIMEX1:	illegal time (greater than 24 hours)
ZONEX1:	illegal time zone

Section 8 Index

PRIVILEGED JSYS'S

INTRODUCTION

This section of the JSYS manual discusses privileged JSYS's which ordinary users need not know about and cannot execute. This is labeled Section 1Ø, intentionally reserving Section 9 for "MISCELLANEOUS".

CRDIR JSYS 24Ø

Creates or modifies a directory entry.

ACCEPTS IN 1: string pointer to directory name

2: LH:
- BØ set directory name from parameter block NOT IMPLEMENTED YET
- B1 set password from parameter block
- B2 set maximum disc file storage from parameter block
- B3 set privilege bits from parameter block
- B4 set mode bits from parameter block
- B5 set special resource information from parameter block
- B6 set directory number from parameter block (only legal for new directories)
- B7 set default file protection from parameter block
- B8 set directory protection from parameter block
- B9 set default backup spec from parameter block
- B1Ø set last log in
- B11 set user group word from parameter block
- B12 set directory group word from parameter block
- B13 not used
- B14 not used
- B15 not used
- B16 delete this directory entry

RH:
pointer to parameter block (E)

3: device designator

CRDIR

RETURNS +1: always, with directory number in 1

Requires WHEEL or OPERATOR status enabled. (This restriction may be removed in part to enable a user to change his own password, decrease maximum disc file storage, etc.) Generates illegal instruction trap on error conditions.

CRDIR ERROR MNEMONICS:

CRDIX1:	WHEEL or OPERATOR special capability not enabled
CRDIX2:	name pointed to by 1 is old and directory number disagrees with it
CRDIX3:	no room in JSB
CRDIX4:	no room in sub index
CRDIX5:	null name not allowed
CRDIX6:	name pointed to by 1 is new but directory number is already used

Parameter Block

E string pointer to directory name
E+1 string pointer to password
E+2 max # pages which can be used for perm disc storage
E+3 B18 user is WHEEL, B19 user has OPERATOR privileges
B2Ø user can see system confidential information
(e.g. error printouts)
E+4 BØ off directory name useable for logging in or CNDIR
on directory name can only be used for CNDIR
B1 off accounts must be numeric
on accounts may be alphanumeric
B2 (used only by EXEC)
off, print each login message only once
on, repeat login message at each login
E+5 special resource info not currently used
E+6 directory number, if user specifies a non-unique number this will generate an error.
E+7 default file protection (18 bits right justified)
E+1Ø directory protection (18 bits right justified)
E+11 default file retention specification. Currently only the right 4 bit field is used to specify the default number of versions of a file to retain on the disc.
E+12 data to set last log in to
E+13 user group word
E+14 directory group word

If CRDIR is for a new directory name, the new directory name is pointed to by 1. If BØ of E+4 is off, a password must be supplied (B1 of 2 ignored). Other bits of 2 may be on or off optionally. In particular, if the following bits are off, CRDIR will default fields to:

B2 off sets maximum disc file storage to 512 pages
B3 off implies no special privileges
B4 off implies directory name which can be used for logging in but requires numeric account
B5 off implies no special resource information
B6 off finds the next highest (but unused) directory number and assigns it to this directory name. B6 should normally be off.
B7 off sets default file protection to 777752
B8 off sets directory protection to 77774Ø
B9 off sets default file retention word to 2
B1Ø off sets last login to 18 November 1858
B11 off sets user group word to Ø
B12 off sets directory group word to Ø

If CRDIR is for an old directory name and if any of BØ-B17 are off, the particular parameters are simply not affected.

GTDIR JSYS 241

Gets directory entry information.

ACCEPTS IN 1: directory number

2: pointer to where remaining "parameters" should be stored (E)

3: string pointer for where to store password

GTDIR

RETURNS +1: always, updated string pointer in 3

Requires WHEEL or OPERATOR status enabled. Generates illegal instruction PSI on error conditions below.

The parameter block at E is compatible with CRDIR. E+Ø is not touched by the JSYS. E+1 points to the beginning of the password string (i.e., the argument supplied in 2). From E+2 to E+14, items are the same as CRDIR. E+12 receives the date and time of last LOGIN.

GTDIR ERROR MNEMONICS:

GTDIX1:	neither WHEEL nor OPERATOR special capability enabled
GTDIX2:	no such directory

DSKOP JSYS 242

Disk operate. Allows hardware-oriented specification of disk transfers. Requires WHEEL or OPERATOR special capability.

ACCEPTS IN 1: If BØ off, right justified
hardware disk address
If BØ on, right justified
linear (software) address.

2: B13: write header
B14: write
B15: compare class
B16-B24: class
B25-B35: word count (max 1ØØØ octal)

3: virtual core address

DSKOP

RETURNS +1: always
1: B24-B35: error bits from 63
Ø if no errors

Transfer must not cross page boundary.

If transfer produces any kind of error, 5 repeats will be attempted, and the error bits from the last attempt returned in 1.

Generates illegal instruction PSI on error conditions below.

DSKOP ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR special capability
not enabled

SPRIW JSYS 243

Set PRIority word. JSYS under development. Requires enabled WHEEL capability.

DSKAS JSYS 244

DiSK ASsignment control, attempts assignment of specific disk address.

ACCEPTS IN 1: disk address

DSKAS

RETURNS +1: unsuccessful, address already assigned

+2: address available and assigned

DSKAS may also be used to deassign an assigned disk address by setting BØ of the disk address word in 1. If the address was assigned, return is to caller plus 1. If the address was not previously assigned, a BUGCHK will occur in the monitor.

DSKAS may also be used to request assignment of a free disk page near the disk address given in 1 by also having B1 of 1 set in the call. Assignment will be on the same track as specified by the disk address if possible or else on a nearby track. If the disk address given is Ø, a page will be assigned on a track which is at least 1/2 free. In this last case of DSKAS, a return to caller plus 1 means the assignment was impossible because the disc was full. The return to caller plus 2 returns the assigned disk address in 1.

Generates illegal instruction PSI on error conditions below.

DSKAS ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR special capability not enabled

SJPRI JSYS 245

Set Job Priority. Sets the scheduler priority control word for any job. Requires WHEEL or OPERATOR capability.

ACCEPTS IN 1: Job number

2: Priority word

SJPRI

RETURNS +1: illegal or non-existent job number
+1: successful completion

The priority word is set in the top fork of the designated job, but not in any existing inferior forks. The priority word is always passed down when a fork is created. A priority word of Ø means no special action. A job may be guaranteed a certain percentage of CPU resources (CPU time/real time) by placing the desired percentage (1-99) in the left half of the priority word.

The right half is used for some installation dependent or experimental purposes and should always be Ø.

SJPRI ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability not enabled.

MSFRK JSYS 312

MSFRK (Monitor Start FoRK) is exactly like SFORK (see Section 6) except that it uses the entire contents of AC2, including the User Mode and other flags, for starting the fork. Thus a fork can be started in monitor mode. This JSYS is legal only if:

a) It is called from monitor mode, or
b) Wheel or Operator capability is enabled.

MSFRK was implemented to allow job zero to spawn multiple forks to handle various asynchronous monitor tasks.

Note: The starting context is undefined, so the fork being started should execute the following sequence at its starting address:

```
FBGN:   MOVSI A,UMODF    ;A FAKE USER MODE PC
        MOVEM A,FPC      ;SIMULATE A JSYS CALL
        JSYS MENTR       ;ESTABLISH USUAL CONTEXT
```

This causes a stack and AC block to be set up, etc.

HSYS JSYS 3Ø7

Accepts in 1: Date and time in standard GTAD format, for scheduled shutdown, or Ø to abort a shutdown in progress.

2: Date and time service will resume, for information only

This JSYS declares a specific time at which the system will be forcefully shut down, and then proceeds to do so. Therefore, it requires WHEEL, OPERATOR, or MAINTENANCE capability. Notification will be broadcast to all users periodically before the shutdown. When the shutdown time arrives, any remaining users will be logged off, except those on the CTY and OTY (see below under startup procedure). The message "SHUTDOWN COMPLETE" will be typed on the CTY, and unless a job is still on one of the two privileged terminals, the machine will halt.

The current shutdown and restart times are available using the GETAB's on the SYSTAT system table (see Section 3).

USRIO JSYS 31Ø

Enables execution of I/O instructions in user mode for maintenance programs. Requires MAINTENANCE, WHEEL, or OPERATOR capability.

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

USRIO ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability not enabled

PEEK JSYS 311

Transfers a block of words from monitor to user space.

ACCEPTS IN 1: XWD word count,
first monitor address

2: first user address

PEEK

RETURNS +1: failed, error number in 1

+2: success, n words transferred.

Block must not cross monitor space page boundary, and MAINTENANCE, WHEEL, or OPERATOR capability is required.

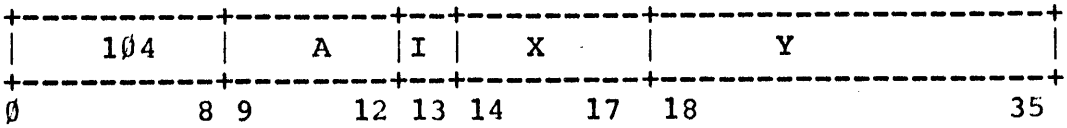
PEEK ERROR MNEMONICS:

CAPX1: (special capability required)
PEEKX1: transfer cannot cross monitor page boundary
PEEKX2: monitor page read-protected

Section 1Ø Index

APPENDIX Index

JSYS Jump to System



If the effective address is 1000 [8] or greater, fetch a double (left half, right half) pointer from the effective address. Store the flags and incremented PC in the location indicated by the left half pointer and jump to the location indicated by the right half pointer. The flags and PC are stored in the left and right halves respectively of the indicated word.

If the effective address is less than 1000 [8], the instruction is an EX JSYS (exec JSYS). Enter monitor mode (leave user mode) and fetch a double pointer from the real core address (unmapped) equal to 1000+ the effective address. As above, the flags and PC are stored through the left half pointer and control jumps (in monitor mode) to the location indicated by the right half pointer.

This instruction provides a very fast way for a user program to call on Monitor-provided services such as word or character I/O, or number conversion. Arguments are typically passed and returned in the AC's, and the Monitor returns control with a JRST 2,@ through the location where the PC and flags were stored.

This instruction may also be used in user mode (with an effective address greater than 1000) to dispatch through a transfer vector into routines which must be kept reentrant. (The pointers for storing PC and flags would point into a temporary storage region in this case.)

LGINX1	600010	LOGIN: Illeg account
LGINX2	600011	LOGIN: Can't Log in under that directory number
LGINX3	600012	LOGIN: NO ROOM in System
LGINX4	600013	LOGIN: Password incorrect
LGINX5	600014	LOGIN: Already Logged in
CRJBX1	600020	CRJOB: Cannot open Pri In file
CRJBX2	600021	CRJOB: Cannot open Pri Out file
CRJBX3	600022	CRJOB: Illeg account
CRJBX4	600023	CRJOB: Terminal not available
CRJBX5	600024	CRJOB: Illeg directory #
CRJBX6	600025	CRJOB: NO ROOM in system
CRJBX7	600026	CRJOB: Password incorrect
LOUTX1	600035	Illegal attempt to LGOUT job
LOUTX2	600036	LGOUT: No such job
CACTX1	600045	CACCT: Illegal account
CACTX2	600046	CACCT: Not logged in
EFCTX1	600050	EFACT: LOG Spec Cap not enabled
EFCTX2	600051	EFACT: Entry longer than 64 words
EFCTX3	600052	EFACT: Accounting system failure,
Contact Operator or Systems Personell		
GJFX1	600055	GTJFN: Desired "JFN" illeg
GJFX2	600056	GTJFN: Desired JFN not available
GJFX3	600057	GTJFN: No JFN's available
GJFX4	600060	GTJFN: Illeg char
GJFX5	600061	GTJFN: More than 39 chars in a field
GJFX6	600062	GTJFN: Two device fields or device not first
GJFX7	600063	GTJFN: Two directory fields or directory after name
GJFX8	600064	GTJFN: "<"not following ">"
GJFX9	600065	GTJFN: More than 1 name field
GJFX10	600066	GTJFN: Non-numeric version
GJFX11	600067	GTJFN: Two version fields
GJFX12	600070	GTJFN: Two accounts
GJFX13	600071	GTJFN: Two protections
GJFX14	600072	GTJFN: Bad protection
GJFX15	600073	GTJFN: Illeg confirmation char
GJFX16	600074	GTJFN: No such device
GJFX17	600075	GTJFN: No such directory
GJFX18	600076	GTJFN: No such file name
GJFX19	600077	GTJFN: No such extension
GJFX20	600100	GTJFN: No such version
GJFX21	600101	GTJFN: No such file (it disappeared)
GJFX22	600102	GTJFN: JSB full
GJFX23	600103	GTJFN: Directory full
GJFX24	600104	GTJFN: Old file required
GJFX25	600105	GTJFN: Name given for non-directory device
GJFX26	600106	GTJFN: Extension for non-directory device
GJFX27	600107	GTJFN: New file required
GJFX28	600110	GTJFN: Device not mounted
GJFX29	600111	GTJFN: Device not available
GJFX30	600112	GTJFN: Numeric account required

GJFX31	600113	GTJFN: Illeg "*"
GJFX32	600114	GTJFN: Empty directory and "*" given for name
GJFX33	600115	GTJFN: NULL NAME NOT ALLOWED
OPNX1	600120	OPENF or RNAMF: file open
OPNX2	600121	OPENF: File doesn't exist
OPNX3	600122	OPENF: Read access not allowed
OPNX4	600123	OPENF: Write access not allowed
OPNX5	600124	OPENF: Execute access not allowed
OPNX6	600125	OPENF: Append access not allowed
OPNX7	600126	OPENF: Device assigned to another job
OPNX8	600127	OPENF: Device not mounted
OPNX9	600130	OPENF: File busy
OPNX10	600131	OPENF: NO ROOM
OPNX11	600132	OPENF: WHEEL or OPER spec cap needed to not change dates
OPNX12	600133	OPENF: List access not allowed
OPNX13	600134	OPENF: Illeg access
OPNX14	600135	OPENF: Illeg mode
OPNX15	600136	OPENF: Non read/write access not allowed
OPNX16	600137	OPENF: File has bad index block
OPNX17	600140	OPENF: NO ROOM FOR LONG FILE PAGE TABLE TABLE
OPNX18	600141	OPENF: NO ROOM TO INITIALIZE INDEX BLOCK
OPNX19	600142	OPENF: IMP IS NOT UP
OPNX20	600143	OPENF: HOST IS NOT UP
OPNX21	600144	OPENF: CONNECTION REFUSED
OPNX22	600145	OPENF: CONNECTION BYTE SIZE MISMATCH
DESX1	600150	Illeg TENEX source/destination designator
DESX2	600151	That terminal not available
DESX3	600152	Unassigned JFN
DESX4	600153	Illeg use of terminal designator or string pointer
DESX5	600154	File not open
DESX6	600155	File must be A TTY
DESX7	600156	JFN MUST NOT REFER TO OUTPUT ASTERISKS
DESX8	600157	FILE MUST BE ON DISK
CLSX1	600160	CLOSF: File not open
CLSX2	600161	CLOSF: File not closable by this fork
RJFNX1	600165	RLJFN: File is open
RJFNX2	600166	RLJFN: JFN being used to accumulate file name
RJFNX3	600167	RLJFN: JFN not accessible to this fork
DELFX1	600170	DELF: You don't have delete access to file
SFPTX1	600175	SFPTR: File not open
SFPTX2	600176	SFPTR: illeg to set this file's pointer
SFPTX3	600177	SFPTR: illeg pointer (eg less than -1)
CNDIX1	600200	CNDIR (used for password check only): Incorrect password
CNDIX2	600201	CNDIR: Incorrect password
CNDIX3	600202	CNDIR: Invalid directory #
CNDIX4	600203	CNDIR: Can't check password when logged in
CNDIX5	600204	CNDIR: Not logged in
SFBSX1	600210	SFBSZ: Can't change byte size for this opening of file
SFBSX2	600211	OPENF or SFBSZ: Illeg byte size
IOX1	600215	Reading from a file not open for reading

```
IOX2     600216  Writing on a file not open for writing
IOX3     600217  RIN or ROUT: Can't change pointer for this opening of fi
                                                        **le
IOX4     600220  Reading beyond end of file
IOX5     600221  Data error
PMAPX1   600240  PMAP: Illeg access request
PMAPX2   600241  PMAP: Illeg case
SPACX1   600245  SPACS: Illeg access request
FRKHX1   600250  Illeg fork handle
FRKHX2   600251  Attempt to manipulate a superior fork
FRKHX3   600252  Illeg attempt to manipulate multiple forks
FRKHX4   600253  Fork running
FRKHX5   600254  HFORK: Fork(s) already halted
FRKHX6   600255  CFORK: Fork handles used up
GTABX1   600267  GETAB: Illeg table index
GTABX2   600270  GETAB: Illeg table #
RUNTX1   600273  RUNTM: Fork handle -3 or -4 given
STADX1   600275  STAD: Date & time already set and not WHEEL or OPER
STADX2   600276  STAD: Ridiculous date or time
ASNDX1   600300  ASND: Unassignable device
ASNDX2   600301  ASND: Illeg to assign this device
ASNDX3   600302  ASND or CSYNO: No such device
CSYNX1   600312  CSYNO: Synonym already in use
ATACX1   600320  ATACH: Illeg job #
ATACX2   600321  ATACH: Job already attached
ATACX3   600322  ATACH: Directory # does not match
ATACX4   600323  ATACH: Password Incorrect
ATACX5   600324  ATACH: No controlling terminal
DCHRX1   600330  DVCHR: Illeg device designator
STDVX1   600332  STDEV: No such device
DEVX1    600335  Illeg device designator
DEVX2    600336  Device assigned to another job
DEVX3    600337  Device not mounted
MNTX1    600345  MOUNT: Bad directory
MNTX3    600347  MOUNT: Unmountable device
TERMX1   600350  ATI or DTI: That code not useable for PSI's
ATIX1    600352  ATI: Illeg channel #
ATIX2    600353  ATI: Spec cap required to assign ^C
DTIX1    600355  DTI: That code wasn't assigned to you
TTYX1    600360  STI: Not a terminal or no such terminal
CFRKX2   600362  CFORK: Can't start fork with no pages
CFRKX3   600363  CFORK: NO ROOM in System
KFRKX1   600365  KFORK: Can't kill top level fork
KFRKX2   600366  KFORK: Can't kill self
RFRKX1   600367  RFORK: Attempt to resume non-frozen fork(s)
GFRKX1   600371  GFRKS: Illeg fork handle in 2
GETX1    600373  GET: Bad save file format
GETX2    600374  GET: System's Special Pages Table full
SFRVX1   600377  SFRKV: Illeg ent vect relative position
NOUTX1   600407  NOUT: Radix out of range 2 to 10
```

```
NOUTX2  600410  NOUT: Column overflow
IFIXX1  600414  NIN: Radix out of range 2 to 10
IFIXX2  600415  NIN: First character not a digit
IFIXX3  600416  NIN: Overflow
GFDBX1  600424  GTFDB: Illeg displacament
GFDBX2  600425  GTFDB: Illeg number of words
GFDBX3  600426  GTFDB: You don't hava list access to file
CFDBX1  600430  CHFDB: Illeg displacament
CFDBX2  600431  CHFDB: Illeg to change those bits
CFDBX3  600432  CHFDB: You don't hava spec cap or file access to change
                                **those bits
CFDBX4  600433  CHFDB: Illeg value for those bits
DUMPX1  600440  DUMPI/O: Command list error
DUMPX2  600441  DUMPI/O: JFN wasn't opened in mode 17
DUMPX3  600442  DUMPI/O: Address too big or crosses end of memory
DUMPX4  600443  DUMPI/O: Memory access error
RNAMX1  600450  RNAMF: Files not on same device
RNAMX2  600451  RNAMF: DESTINATION FILE DISAPPEARED
RNAMX3  600452  RNAMF: ACCESS TO DESTINATION NOT ALLOWED
RNAMX4  600453  RNAMF: NO ROOM
BKJFX1  600454  Can't BKJFN that device twice
TIMEX1  600460  Time greater than 24 hours
ZONEX1  600461  Time zone out of range -12 to 12
ODTNX1  600462  ODTIM can't print time zones except USA and Greenwich
DILFX1  600464  Illegal date format
TILFX1  600465  Illegal time format
DATEX1  600466  Year out of range 1858 to 2576
DATEX2  600467  Month greater than 11
DATEX3  600470  Day of month too large
DATEX4  600471  Day of week greater than 6
DATEX5  600472  Date out of legal range
DATEX6  600473  System date and time not set
SMONX1  600515  SMON: LOG spec cap not enabled
CPRTX1  600520  CPRTF: You don't hava access to file to change protectio
                                **n
SACTX1  600530  SACTF: File not on multiple-directory device
SACTX2  600531  SACTF: JSB full
SACTX3  600532  SACTF: Directory requires numeric account
SACTX4  600533  SACTF: You don't hava access to file to change account
GACTX1  600540  GACTF: File not on multiple-directory device
GACTX2  600541  GACTF: File doesn't axist
FFUFX1  600544  FFUFP: File not open
FFUFX2  600545  FFUFP: File not on multiple-directory device
FFUFX3  600546  FFUFP: No used page found
DSMX1   600555  DSMNT: Can't dismount -- file(s) open
RDDIX1  600560  RDDIR failure
SIRX1   600570  SIR: table address less than 20
SSAVX1  600600  SSAVE: File not on disk
SSAVX2  600601  SSAVE: Page count greater than 1000
SEVEX1  600610  SEVEC: Entry vector longer than 777
```

```
WHELX1   600614  WHEEL or OPERATOR special capability not enabled
CRDIX1   600620  CRDIR: WHEEL or OPER spec cap not enabled
CRDIX2   600621  CRDIR: Wrong directory # for name
CRDIX3   600622  CRDIR: JSB full
CRDIX4   600623  CRDIR: Sub-index full
CRDIX5   600624  CRDIR: Null name illeg
CRDIX6   600625  CRDIR: Can't change # of old directory
CRDIX7   600626  CRDIR: CAN'T KILL DIRECTORY WITH BUSY FILES
GTDIX1   600640  GTDIR: WHEEL or OPER spec cap not enabled
GTDIX2   600641  GTDIR: No such directory #
FLINX1   600650  FLIN: First char not blank or numeric
FLINX2   600651  FLIN: Number too small
FLINX3   600652  FLIN: Number too large
FLINX4   600653  FLIN: Bad format
FLOTX1   600660  FLOUT: Field overflow before or after point
FLOTX2   600661  FLOUT: Exponent field overflow
FLOTX3   600662  FLOUT: Illeg format spec
FDFRX1   600700  FDFRE: Not a multipla directory device
FDFRX2   600701  FDFRE: No such directory #
ATPX1    600710  ATPTY: Bad rec'v JFN
ATPX2    600711  ATPTY: Rec'v JFN not read
ATPX3    600712  ATPTY: Rec'v JFN not open
ATPX4    600713  ATPTY: Rec'v JFN not net
ATPX5    600714  ATPTY: Rec'v JFN not unused
ATPX6    600715  ATPTY: Rec'v connection refused
ATPX7    600716  ATPTY: Send JFN Bad
ATPX8    600717  ATPTY: Send JFN not write
ATPX9    600720  ATPTY: Send JFN not open
ATPX10   600721  ATPTY: Send JFN not net
ATPX11   600722  ATPTY: Send JFN not unused
ATPX12   600723  ATPTY: Send connection refused
ATPX13   600724   ATPTY: No NVT's
CVSKX1   600730  CVSKT: Bad JFN
CVSKX2   600731  CVSKT: Local socket illegal in this context
DPX1     600734  DISPLAY JSYS: UNASSIGNED DISPLAY PROCESS
DPX2     600735  DISPLAY JSYS: ILLEGAL DISPLAY PROCESS NUMBER
STRDX1   600740  STRDP (STSDP): ILLEGAL MAP WORD
STRDX2   600741  STRDP (STSDP): TOO MANY PAGES LOCKED
STRDX3   600742  STRDP (STSDP): NON-READABLE-WRITABLE-EXECUTABLE PAGE
STTX1    600744  STTYP: Illegal terminal type number
RNAMX5   600750  RNAMF: DESTINATION BUSY
RNAMX6   600751  RNAMF: DESTINATION HAS BAD PAGE TABLE
RNAMX7   600752  RNAMF: SOURCE DISAPPEARED
RNAMX8   600753  RNAMF: ACCESS TO SOURCE NOT ALLOWED
RNAMX9   600754  RNAMF: SOURCE IS EMPTY
RNMX10   600755  RNAMF: SOURCE BUSY
RNMX11   600756  RNAMF: SOURCE HAS BAD PAGE TABLE
RNMX12   600757  RNAMF: RENAME TO SELF
ILINS1   600770  Illegal instruction executed
ILINS2   600771  Unassigned JSYS executed
ILINS3   600772  10/50 UUO executed and compatability file not found
```

JSYS MASTER INDEX

1 September 1973

The following pages contain listings of the JSYS's, alphabetically and numerically, with the section in which the JSYS appears and the corresponding page number.

JSYS MASTER INDEX

1 September 1973

Bolt Beranek and Newman Inc.

50 Moulton Street
Cambridge, Massachusetts 02138
(617) 491-1850

NIC No. 19031