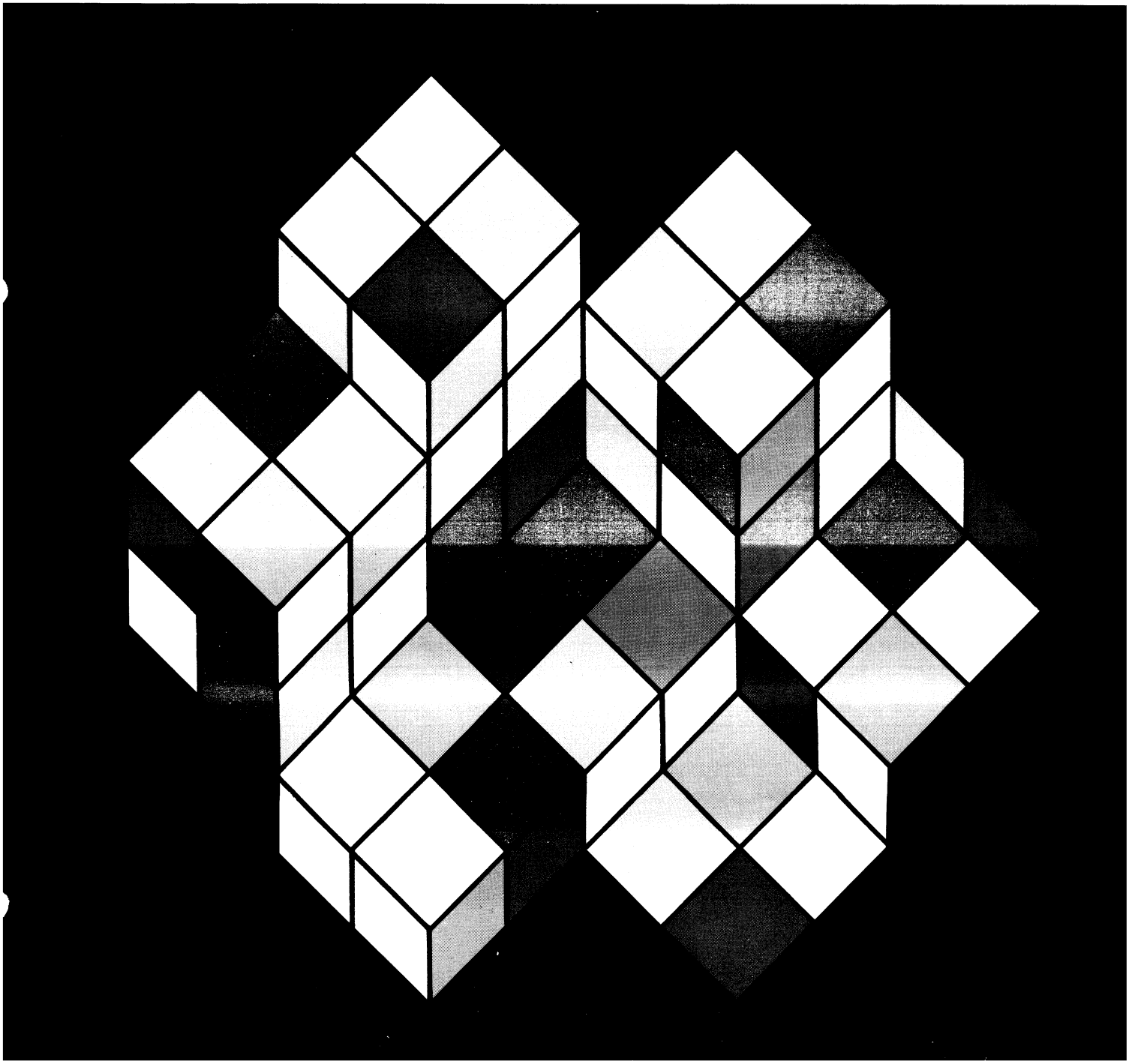
Bolt Beranek and Newman Inc.

TENEX

NIC No. 16874

Executive Manual



TENEX EXECUTIVE LANGUAGE

MANUAL FOR USERS

Theodore H. Myer
John R. Barnaby

Revised, April, 1973
William W. Plummer

Bolt Beranek and Newman Inc.
5Ø Moulton Street
Cambridge, Massachusetts Ø2138

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
A.	SAMPLE DIALOGUE	2
II.	EXECUTIVE LANGUAGE STRUCTURE	6
A.	SIGNALING TENEX	6
1.	(CTRL)-C	6
2.	@ - The TENEX Ready Symbol	7
3.	; - The TENEX Comment Symbol	7
B.	TENEX COMMAND STRUCTURE	7
1.	Command Components	7
2.	Command Fields	8
3.	Command Input and Recognition	8
4.	Command Terminators	10
5.	Examples	11
6.	Default Values and Null Values	12
7.	Subcommands	15
8.	Format Options	15
C.	EDITING AND ERRORS IN TENEX COMMANDS	17
1.	(CTRL)-A	17
2.	(CTRL)-W	18
3.	(CTRL)-R	18
4.	(CTRL)-X, Rubout	18
5.	Errors	18
D.	FORMAT CONVENTIONS IN MANUAL DESCRIPTIONS	20
III.	SYSTEM ACCESS	21
A.	GENERAL INFORMATION	21
1.	User Names	21
2.	Passwords	21
3.	Accounts	22
4.	Jobs	22
B.	ACCESS COMMANDS	23
1.	LOGIN	23
2.	LOGOUT	25
3.	CHANGE	26
4.	DETACH	26
5.	ATTACH	28

Page

IV. TENEX FILE SYSTEM....30

A. TENEX FILE DESIGNATORS....30

1. Device....31
2. Directory Name....32
3. File Name....33
4. Extension....34
5. Version Number....35

B. TYPING FILE DESIGNATORS....36

1. Relevant Descriptors....37
2. Default Values....37
3. Recognition....38
4. Examples....39

C. MULTIPLE FILE DESIGNATORS....40

1. File Designator Lists....40
2. The * Feature....40

D. OPTIONAL DISK FILE DESCRIPTORS....41

1. Temporary Files....42
2. File Protection....42
3. Directory Protection....44
4. Groups....44
5. Account....45
6. Examples....45

E. FILE DESIGNATOR ERRORS....46

F. FILE COMMANDS....47

1. DIRECTORY....48
2. QFD....52
3. COPY....52
4. APPEND....57
5. RENAME....58
6. ACCOUNT....59
7. PROTECTION....59
8. DELETE....59
9. UNDELETE....60
10. EXPUNGE....61
11. CLEAR....61
12. SHUT....62
13. JFNCLOSE....62
14. CONNECT....62
15. LIST....63
16. TYPE....65
17. ARCHIVE....66

Page

18. INTERROGATE....67
19. EPHEMERAL....67
20. NOT EPHEMERAL....68

G. FUTURE ADDITIONS....68

V. DEVICE HANDLING....69

A. DEVICE COMMANDS....69

1. ASSIGN....70
2. DEASSIGN....70
3. MOUNT....70
4. UNMOUNT....71
5. REWIND....71
6. UNLOAD....71

B. FUTURE ADDITIONS....72

VI. SUBSYSTEM CONTROL....73

VII. PROGRAM CONTROL AND DEBUGGING....77

A. PROGRAM CONTROL AND DEBUGGING COMMANDS....78

1. SAVE....78
2. SSAVE....79
3. ENTRY VECTOR....80
4. GET....81
5. MERGE....82
6. RUN....83
7. START....84
8. REENTER....84
9. GOTO....84
10. CONTINUE....85
11. RESET....85
12. Examine Location....86
13. Deposit into Location....86
14. DDT....87
15. IDDT....88
16. NO IDDT....88
17. FORK....88
18. EXEC....89
19. QUIT....89
20. EDIT....90

B. FUTURE ADDITIONS....90

1. Environment (.ENV) Files and the Dump Command....90

Page

VIII. QUERIES....92

1. The ? Feature....92
2. AVAILABLE LINES/DEVICES....93
3. DAYTIME....93
4. WHERE (IS USER)....93
5. SYSTAT....94
6. VERSION....95
7. JOBSTAT....95
8. RUNSTAT....96
9. FILSTAT....96
10. MEMSTAT....97
11. USESTAT....98
12. DSKSTAT....98
13. PISTAT....99
14. FORKSTAT....99
15. Control-T....99

IX. TERMINAL CHARACTERISTICS COMMANDS....101

1. Half or Full Duplex....101
2. Other Terminal Features....101
3. STOPS....101
4. RAISE....102
5. INDICATE....102
6. Padding....103
7. WIDTH....103

X. TERMINAL LINKING AND ADVISING....104

1. LINK....104
2. ADVISE....105
3. BREAK....106
4. REFUSE....106
5. RECEIVE....107

XI. ARPANET COMMANDS....108

1. NETLOAD....108
2. Future Additions....108

XII. FUTURE COMMAND GROUPS....109

1. Input/Output Redirection....109

INDEX....110

I. INTRODUCTION

TENEX is a time-sharing system, built around the DEC PDP-1Ø computer. This manual describes the TENEX Executive Language, the primary vehicle through which users communicate and work with TENEX.

TENEX offers the user three distinct facilities. First, TENEX contains a number of subsystems, entities each of which does a particular job. Some subsystems, such as TECO and BBN-LISP, provide complete computation services, are highly self-contained, and require little knowledge of the remainder of TENEX or of the PDP-1Ø computer. Other subsystems do specific jobs such as editing or compilation, are typically used together with other subsystems, and require of the user more complete knowledge of TENEX. The properties and communication languages for the subsystems are contained in the TENEX User's Guide and in separate subsystem manuals.

Distinct from its subsystems, TENEX offers an entity, known as a virtual computer, that gives the user a vehicle for running machine language programs. This entity is termed a "computer" because it has all the appearance of a piece of computing hardware, "virtual" because some of this appearance is in fact supplied by the system software that controls and drives TENEX. In particular, the virtual computer provides what appears to be more powerful core memory structure and input-output system than actually exists in the PDP-1Ø hardware. The language of the TENEX virtual computer contains a subset of the PDP-1Ø machine language, together with a series of calls on the TENEX software system, known as "JSYS's" and documented in the TENEX JSYS Manual.

The third facility offered by TENEX is a file system that also provides access to and control over the various input and output devices in the system. TENEX files can be kept on disk, magnetic tape, and DECtape. The paper tape reader and punch and the line printer are also treated as files, and terminals can, if desired, be accessed through the file system. The ARPA network connections also look like files to the user. This manual describes the TENEX file system in sufficient detail for most uses. Access to files through TENEX machine language programs is covered in the JSYS Manual.

The TENEX Executive Language allows the user to access and control these three facilities. The language consists of a series of commands that can be entered from a keyboard terminal. It is also possible to control TENEX operations through commands stored in files, although this mechanism is

not in finished form.

The next section of this manual discusses the structure of TENEX commands and other conventions of the Executive Language. Subsequent sections describe separate groups of commands that provide access to the system, perform file and input/output functions, and provide access to the TENEX subsystems. Following sections cover commands that provide program control and debugging, provide information about the system, and communicate information about terminal characteristics to TENEX.

This manual describes a particular version of the TENEX Executive Language. Some of the sections noted above will have further commands in future versions, and these are described briefly at the end of each section. Some new groups of commands will also be added, and these are described at the end of the manual.

A. SAMPLE DIALOGUE

The following illustrates a typical dialogue with TENEX, involving the preparation and running of a user program. In the dialogue, characters typed by the user are underlined; the remaining text is typed by TENEX. Carriage return is shown by %, altmode by $. Comments to the right of the typescript explain the action and reference sections in the manual where the various TENEX commands are described in detail. Prior to the beginning of the dialogue, the user is assumed to have connected his terminal to the computer, either through the telephone system, or by direct patching.

```
BBN-TENEX 1.31.5, BBN-SYSTEM-A EXEC 1.5Ø.34
```

The user signals TENEX with (CTRL)-C (II-A-1). Tenex responds with a standard message.

```
@LOG$IN (USER) JONES$ (PASSWORD)$ (ACCOUNT #) 11451%
JOB 4 ON TTY24 13-APR-73 16:52
```

The user logs in to the system (III-B-1). The @ character means TENEX awaits a command. Altmode causes TENEX to "recognize" (II-B-3) and type back certain portions of commands and arguments.

```
@AS$SIGN DTA2:%
@COP$Y DTA2:ALPHA.MAC (TO) ALPHA.MAC;1 [NEW FILE]%
@DEA$SSIGN DTA2:%
```

Having assigned (V-A-2) DECtape 2, the user copies a file from it onto the disk (1V-F-3), and then releases the DECtape (V-A-1).

```
@DIR$ECTORY%

 ALPHA.MAC;1
 BETA.MAC;2
    .REL;1
@
```

The user lists his file directory (III-F-1). Prior to loading ALPHA, the disk already contained two versions of file BETA.

```
@TEC$O.SAV;1Ø%
*;Y$
INPUT FILE: A$LPHA.MAC;1%
  .
  .
*;U$
OUTPUT FILE: $ALPHA.MAC;2 [NEW VERSION]%
*↑C
```

The user calls in the TECO editing subsystem (VI-2). Using commands described in the TECO manual, he brings in the file previously copied to disk, and, after editing (not shown here) writes a new version (IV-A-5) onto the disk. (CTRL)-C returned control to TENEX.

```
@LIST$ (FILE) A$LPHA.MAC;2%
```

The user requests a line printer listing of the newly edited file (IV-F-15).

```
@MAC$CRO.SAV;7%
*ALPHA←ALPHA%
  .
  .
*↑C
@
```

The MACRO-1Ø assembler (VI-4) is called in to produce a relocatable binary version of ALPHA. TENEX will automatically assign a .REL extension (IV-A-4) to the output file. Messages typed by MACRO were omitted for clarity. As before, (CTRL)-C forced a return to TENEX.

```
@LOADER$.SAV;42%
*ALPHA%
*BETA%
*$
  .
  .
@
```

The user calls in the LOADER subsystem (VI-6) to link- load the binary versions of ALPHA and BETA into TENEX core. (LOADER messages were omitted). The LOADER returns control automatically to TENEX.

```
@SAV$E (CORE FROM) $2Ø (TO) $777777 (ON) &
 PROG.SAV;1  [NEW FILE]%
```

All assigned core memory is saved (VII-A-1) in file PROG. In this example the line-continuation character & was used. This is described fully in another section of this manual.

```
@ST$ART%
  .
  .
  .
```

The program is started (VII-A-7)

```
@DIR$ECTORY%

 ALPHA.MAC;2,1
    .REL;1
 BETA.MAC;2
    .REL;1
 PROG.SAV;1
@
```

Subsequent to the program run, a second directory listing confirms three new files - ALPHA.MAC;2 (created through TECO), ALPHA.REL;1 (created by MACRO), and PROG.SAV;1 (created by the SAVE command).

```
@DEL$ETE A$LPHA.M$AC;1%
```

The first, now obsolete, version of ALPHA is deleted (IV-F-8).

```
@LOGO$UT%
KILLED JOB 4, USER JONES, ACCT 11451, TTY 24 AT 4/13/73
 17:52
  USED Ø:3:21 IN Ø:59:46
```

Finally, the user logs out (III-B-2). TENEX types a final message that informs the user, among other things, that he has used 3 minutes, 21 seconds of CPU time during a session that lasted 59 minutes, 46 seconds.

II. EXECUTIVE LANGUAGE STRUCTURE

Communication with TENEX takes the form of a dialogue in which the user gives a command, TENEX performs the desired action, and then waits for a new command. The collection of available commands, together with certain special characters and conventions, makes up what is known as the TENEX Executive Language. This section describes the general form of the language; subsequent sections cover the individual commands.

A. SIGNALING TENEX

1. (CTRL)-C

To signal TENEX hold down the control (CTRL) key and strike "C". At the beginning of a session (CTRL)-C will get the initial attention of TENEX. TENEX will respond with an identifying message followed by the "@" character on the next line:

```
BBN-TENEX 1.31.5, BBN-SYSTEM-A EXEC 1.5Ø.34
@
```

The code numbers in the message identify the version of TENEX in use and will change as the system develops over time. At this point the user can LOGIN as described in Section III-B-1.

After initial contact has been established, (CTRL)-C will continue to get TENEX's attention during the ensuing session. During a session, TENEX will respond to (CTRL)-C by typing "↑C", followed by "@" on the next line. For brevity, the abbreviation ↑C has been used for (CTRL)-C throughout the remainder of this manual.

TENEX will respond to ↑C no matter what is happening at the time. Thus, ↑C will interrupt a running program or file operation, abort a partially typed or partially executed TENEX command, and in all cases control will return to TENEX.

The only partial exception to this rule occurs when ↑C is struck during output on the user's terminal. TENEX will continue the output until the user's output buffer* is empty before responding to a single ↑C, but will respond immediately to a second ↑C.

2. @ - The TENEX Ready Symbol

The "@" typed after ↑C indicates that TENEX is ready for a new command. TENEX will also type "@" on completion of a command, or after aborting a command with one of the editing characters discussed in II-C below.

3. ; - The TENEX Comment Symbol

In most cases typing a semicolon causes the rest of the command line to be considered as a comment rather than a command. This feature is especially useful when two users are conversing using the terminal "LINK" feature. The following illustrates the use of semicolon:

```
@;THIS IS A COMMENT, WHICH WILL BE IGNORED BY THE EXEC
@TECO;  teco will be called at the next end of line
*
```

The asterisk in the above example was TECO's ready character.

B. TENEX COMMAND STRUCTURE

1. Command Components

TENEX commands are made up of three components: keywords, arguments, and noise words. Each command begins with an initial keyword that identifies its main function; it may contain further keywords that select options. Arguments, such as user or file designators, core addresses, or account numbers tell what a command is to act on, or provide values for use in command execution. Noise words (always enclosed in parentheses) make TENEX commands more understandable to the user, but have no effect on command execution.

---

*The output buffer is a holding area within TENEX where characters are stored prior to transmission to the user's terminal.

In the following examples, keywords are indicated by K, arguments by A, and noise words by N.

```
@   CHANGE      (ACCOUNT # TO)     123456
      K               N               A

@   AVAILABLE       LINES
      K               K

@ RENAME  (EXISTING FILE)  ABC.FOO;1 (TO BE) ALPHA.A;1
    K            N             A       N        A
```

2. Command Fields

For purposes of the following discussion it is convenient to think of TENEX commands as split up into fields. Each keyword or argument that the user must specify begins a field; the field runs up to the next keyword or argument or to the end of the command. Thus, a TENEX field contains one argument or one keyword, followed possibly by one or more noise words. The value of a command field is taken to be the particular choice of keyword or argument that begins it. The three commands shown above are repeated below with vertical lines showing their breakup into fields.

```
@ CHANGE (ACCOUNT # TO) 123456

@ AVAILABLE LINES

@ RENAME (EXISTING FILE) ABC.FOO;1 (TO BE) ALPHA.A;1
```

The values of the three fields in the last example are RENAME, ABC.FOO;1, and ALPHA.A;1 respectively.

3. Command Input and Recognition

TENEX allows three styles of input. They can be intermixed freely within a command with a separate style applied, if desired, to each field.

a. Full Input. The user can type any field in its entirety, spelling out fully the keyword or argument and the noise words (if any) with their enclosing parentheses. This style is laborious, but "safe" for the user not yet sure of the language.

The following two input styles apply to fields that begin with keywords. Depending on the circumstances (as described below) they may apply in part or in full to fields that

begin with arguments.

b. Abbreviated Input. The user can omit the noise words altogether from a TENEX command field. He can shorten the keyword by typing at least enough initial characters (terminated by space) to distinguish it from the other keywords acceptable in that context. If the user enters insufficient characters for unique identification, TENEX will type "?" and abort the command. Entering more than the minimum abbreviation is perfectly acceptable. This mode allows experienced users to type commands very concisely.

c. Recognition Input. The user can enter initial keyword characters as above, but terminate with altmode* instead of space. In this event TENEX will type back the remainder of the keyword together with any immediately following noise words - i.e., everything through the end of the current field. Recognition input is helpful to less than fully experienced users because the material typed by TENEX verifies that the intended command has been given and also supplies clues as to what is expected next. (The ? feature, VIII-1, is also helpful in feeling one's way through a TENEX command.)

With altmode, insufficient characters cause much less trouble. TENEX merely rings the terminal bell and waits for more input. As above, more than the minimum characters are perfectly acceptable.

Terminating input at any point in a field beyond the minimum required for recognition will cause TENEX to type back the rest of the field. This is true even when input stops, say, midway through a noise word, and is useful, for example, when the user wants to enter keywords in full but wishes TENEX to supply noise words for clarity.

With certain frequently used keywords, fewer than the minimum characters are accepted by convention in either Abbreviated or Recognition input. These are cited as they occur. An example is LOG which is taken to mean LOGIN even though LOGOUT also begins with LOG.

The following examples show full, abbreviated and recognition input as applied to a command field. In these and all following examples user input is underlined, and altmode is indicated by "$". (The "$" is not printed by TENEX.)

---

* "ESC" on some terminals, "PREFIX" on some others

Full:	RENAME (EXISTING FILE)
Abbreviated:	RE
Recognition:	RE$NAME (EXISTING FILE)

d. Argument Input. TENEX arguments cannot be abbreviated. However any noise words in the field with an argument can be omitted. Thus, the user can partially abbreviate argument fields by typing the argument in full (terminated by space) and then proceeding immediately to the next field.

In certain cases, TENEX will perform recognition as described above on arguments. Whether or not this is possible depends on the argument and the context. For example, account numbers are never recognized, user names are recognized in some contexts, and certain parts of file designators are always recognized. The various cases of argument recognition are cited individually as they occur in the following sections. In particular, file designators have their own set of recognition rules, described in detail in section IV-B. Whether or not an argument is recognized, one can always have TENEX supply the noise words of an argument field by typing the argument in full and terminating with altmode.

The following examples show the three styles of input as applied to a field that begins with a file designator.

Full:	ABC.FOO;1 (TO BE)
Partially Abbreviated:	ABC.FOO;1
Recognition:	AB$C.FOO;1 (TO BE)

4. Command Terminators

Space, altmode, and carriage return can terminate commands, though carriage return is the standard TENEX command terminator. In the general case, once all fields of a command have been entered (and any "recognized" information typed back) TENEX will await a final, confirming carriage return before executing the command. There are the following exceptions:

a. C.R. Terminates Last Field. With most commands, TENEX will begin execution immediately if the user terminates the last field with carriage return instead of space or altmode. As with space or altmode enough characters must have been entered to identify the keyword or argument; as with space, the remainder of the field is not typed back.

b. Confirming C.R. Needed. Some commands, notably those that change or destroy information always require a final

carriage return, even if carriage return ended the last field. The carriage return that terminates such a field is echoed as space. With these commands, TENEX reminds the user that final confirmation is needed by typing a message in square brackets after the last field is entered.

c. No Terminator Needed. With certain "harmless" commands, such as queries about the system, no terminator is required; TENEX will take action as soon as the last field is entered.

d. Subcommands To Follow. Certain TENEX commands can be followed by optional subcommands (see 7 below). Typing comma before a terminating carriage return indicates that subcommands are to follow.

5. Examples

The following examples illustrate some of the characteristics of TENEX command structure, recognition, and termination. Carriage return is indicated by "%"; as before user input is underlined and altmode is indicated by "$".

```
@AVAILABLE LINES
@AV L
@AV$AILABLE L$INES
```

The above are equivalent and represent full, abbreviated, and recognition input as applied to both fields of the command. This system query requires no confirmation; execution would begin after the final space in the first two examples, the final altmode in the third.

```
@AV$AILABLE D$EVICES
@AV L$INES
@AV$AILABLE D
@AV LINES
```

The AVAILABLE command has two options; LINES or DEVICES as selected by the second keyword. As indicated above L or D is enough for TENEX to determine which option was intended. Also, these examples show some combinations of different input styles within a single command.

```
@CH$ANGE (ACCOUNT # TO) 12345 %
@CH 12345%
```

This command requires a confirming carriage return which can follow the last field (first example) or terminate the last

field (second example). The argument "account number" cannot be recognized; it must always be entered in full.

```
@REN$AME (EXISTING FILE) AB$C.FOO;1 (TO BE)
         ALPHA.A;1 [NEW FILE] %

@REN ABC.FOO;1 ALPHA.A;1% [NEW FILE] %
```

RENAME requires a final confirming carriage return regardless of how the last field was terminated (space in the first example, carriage return in the second). In the first example recognition style input was used to enter the first two fields. In the second example abbreviated input was used to enter the first field, which began with a keyword. Abbreviation is never possible with arguments, however, even those that can be recognized. Consequently the argument ABC.FOO;1 was entered in full (it could also have been recognized). It was possible, however, to proceed immediately to the last field ALPHA.A;1, omitting the noise words (TO BE).

6. Default Values and Null Values.

a. Default Values. With certain command fields, TENEX sets aside a default value to be supplied automatically if the user declines to specify a particular value himself. For example, in the AVAILABLE command, LINES is the default value for the second field. There are two ways to trigger the selection of default values. Entering altmode as the first character of a field will cause TENEX to select and print the field's default value, together with any following noise words.

Where more than one defaultable field occurs in a command, these fields are generally grouped together at the end of the command. Entering carriage return as the first character of a field will cause TENEX to default it and any succeeding defaultable fields, but without printing further information. Terminating a field with carriage return will default any further defaultable fields. In either of these cases TENEX behaves as though the carriage return terminated the last field: if the command can be executed without a final carriage return, TENEX will proceed to do so. The following four examples are all equivalent:

```
@AV$AILABLE L$INES
@AV$AILABLE $LINES
@AV$AILABLE %
@AV%
```

b. Default Command If no command is input, TENEX attempts to treat the name as that of a .SAV file to be RUN (See VII. 6.). First, the file is looked for in the directory <SUBSYS>, then in the user's connected directory, and finally, in the user's login directory. Thus, if user Jones is logged in and connected to directory <SMITH>, and types,

@DEMO

the EXEC behaves as if he had typed,

@RUN <SUBSYS>DEMO.SAV

If there is no such program, the EXEC proceeds to simulate

@RUN <SMITH>DEMO.SAV

If the DEMO program cannot be found in the directory SMITH either, the final attempt is to act as if the following was typed:

@RUN <JONES>DEMO.SAV

Last, if the file still cannot be found, a question mark is typed, indicating an error.

This "Default RUN" command feature makes programs in a user's private directories appear as subsystems. They are invoked by simply mentioning the appropriate name.

The scope of the search described above can be limited by supplying a specific directory name as well as the file name.

@<HACKS>CHESS

will run the CHESS program without being connected to the <HACKS> directory. No other directories will be searched if that specific file does not exist.

If user Jones has written a program with a name matching that of a subsystem, he must use the RUN command to run it since just typing its name would find the file in <SUBSYS> first.
NOTE: If the name of the file to be run is typed in completely, a space will be sufficient to confirm the name. This allows the program to process the rest of the command line in any way it pleases. On the other hand, if file name recognition is invoked by using an alt. mode character in the name, the program name must be confirmed with a carriage return.

Examples:

```
@TEC$O.SAV;47 %
*
```

In this example, recognition was used, and the confirming carriage return required. Since TECO begins by typing carriage return and linefeed before the asterisk, a blank line was formed in the type out.

```
@TECO
*
```

In this case, the subsystem name was typed completely and terminated with a space. No confirmation was needed and TECO's ready character appeared on the next line.

c. Null Values. Sometimes a reasonable choice for field value is no value at all, for example in the case of selecting none of several possible options. TENEX allows for this situation by providing for null values. Null is indicated by the character "-". To enter a null value for a field, type "-" followed by one of the usual field terminators (space, altmode, or carriage return). Subsequent interaction with TENEX will take place as though an ordinary field value had been entered.

Generally, when null is one of the possible choices for a field, it will also be the default value, and can be selected by any of the default mechanisms described above. When a field is defaulted to null by carriage return, nothing is printed. When defaulted by altmode, "-" is printed followed by any noise words.

The following examples illustrate a command* with three fields following the initial keyword field, each of which has null as a default value. In the first case, each of the fields has been given a definite value; the other cases show various ways of nulling one or more of the fields in question.

```
@DET$ACH (INFILE) AB$C.FOO;1 (OUTFILE) XYZ.A;1$
         (AND) S$TART%
@DET$ACH (INFILE) $- (OUTFILE) $- (AND) S$TART%
@DET$ACH (INFILE) - - S$TART %
@DET$ACH (INFILE) %
@DET%
```

---

*The DETACH command as shown here is not fully implemented in the 4/13/73 version of TENEX.

7. Subcommands

Argument and keyword fields can provide for some variation in command execution. However, with certain commands (such as DIRECTORY - IV-F-1), so many options are available that the command would become unworkably long if each was assigned a command field. To take care of this situation, TENEX provides for subcommands that modify or add to the effect of the command they follow.

As indicated in 4-d above, typing comma just before terminating a command indicates that subcommands are to follow. TENEX signifies its readiness to accept a subcommand by typing "@@" instead of the customary "@".

Subcommands have the same structure as main commands, and follow the same rules for input style, termination, default and null field values. Generally, when a command takes subcommands, several are provided to specify non-exclusive options. In this case, terminating a subcommand will cause TENEX to store away the information if contained and await another subcommand on the next line. A special subcommand will cause execution of the now modified main command.

A command that takes subcommands can be executed without subcommand input by terminating it in any of the usual ways, rather than by comma. In this case, TENEX will begin execution immediately, assuming a set of default conditions for the various options.

In the following example, a main command, DIRECTORY, is modified by a series of subcommands. The final subcommand, BEGIN, causes execution to start.

```
@DIRECTORY,%
@@LENGTH (IN BYTES)%
@@TIMES (AND DATES OF) CREATION%
@@DOUBLESPACE%
@@BEGIN%
```

8. Format Options

The TENEX Executive language provides a number of options for enhancing and controlling the appearance of TENEX commands. The main intent of these is to assist in future versions of the system, when it will be possible to "drive" TENEX with pre-typed commands stored in TENEX files, but the options can be used at present with directly executed commands.

a. Comments. A comment can be added to a TENEX command by typing ";" instead of carriage return after completing the last field. Text typed at this point will appear in the typescript, but will not affect command execution. Terminating the comment with carriage return will cause execution to begin.

@CH$ANGE (ACCOUNT #TO) 21345;MONEY LOSING OVERHEAD%

b. Spaces and Tabs. Wherever the context will permit (in particular before or after keywords or arguments) spaces and tabs* can be inserted into TENEX commands without affecting command execution, abbreviation or recognition.

@ CH$ANGE (ACCOUNT # TO) 15432 ;CONSULTING%

c. Continuation. A command can be continued on the next line by typing "&" wherever space is legal (except not in terminating a file designator). TENEX will type a carriage return, and the user can continue typing the command on the next line.

@REN$AME (EXISTING FILE) AB$C.FOO;1 (TO BE) &
ALPHA.A;1 [NEW FILE] %

d. Form Feed. Form feed (typed by (CTRL)-L and abbreviated ↑L) can be used as a substitute for carriage return. This will advance the terminal paper to the top of a new page prior to command execution.

e. Lower Case Letters. Lower case letters, when available on a TENEX terminal, may be substituted freely for upper case letters in typing TENEX commands.

---

*Commands relevant to tabulation are described in section IX. On many terminals (CTRL)-I will cause program driven tabulation, with tab stops adjustable through the STOPS command.

C. EDITING AND ERRORS IN TENEX COMMANDS

Occasionally the user will make typographical or other errors in typing a TENEX command. To deal with these situations, TENEX provides a number of special characters that allow the user to edit commands as he types them in. In addition, TENEX itself checks commands for syntactic errors, erroneous keywords or arguments, and system conditions that would prevent command execution. If the user fails to correct an error of this sort in time, TENEX will type a message, and take action appropriate to the error in question.

As the user types a command, TENEX continuously monitors the incoming character stream, decoding each keyword or argument as soon as the user terminates it. This has the benefit of allowing TENEX to detect and cope with errors as soon as they are made. On the other hand it restricts the user's ability to edit a partially typed TENEX command. In particular, once a keyword or argument* has been terminated (with space, altmode, or carriage return), it is no longer available for modification. The result of this restriction is to limit the range of the TENEX editing characters, as noted in the descriptions that follow.

1. (CTRL)-A

To delete a character, type (CTRL)-A (abbreviated ↑A). TENEX will type "\" followed by the character deleted. Successive ↑A's will delete successive characters back to the beginning of the keyword, argument, or noise word group that one is currently typing.

However, ↑A cannot delete back into a previous field, or, if one is typing noise words, back into the preceding argument or keyword. In addition, with multiple element arguments such as file designators, ↑A will delete back to the beginning of the current element but no further. In all of these cases, typing too many ↑A's will cause TENEX to ring the terminal bell, but take no further action.

@AB\BV$AILABLE DEV\V\E\DL$INES

In the above, ↑A was struck before each appearance of \ .

Note: During password input, ↑A will echo only a \ for each character deleted.

---

*or any one element in a multi-element argument.

2. (CTRL)-W

To delete all of the keyword, argument (or argument element) or noise word groups that one is currently typing, strike (CTRL)-W (↑W). TENEX will type "←". Typing ↑W is exactly like typing as many ↑A's as are legal in the current context. Consequently, one can never type more than one ↑W in a row; an attempt to do so will ring the terminal bell.

```
@ABAIL←AVAIL$ABLE DCVICE←DEVICES
```

3. (CTRL)-R

Occasionally, too much editing will so foul up the typescript that one can no longer tell what it means. To take care of this, (CTRL)-R (↑R) typed anywhere in a command, will cause TENEX to carriage return, and then retype the command with all editing carried out, up to the point where ↑R was struck.

```
@CHX\XANHE\E\HGE(ACX\XC←ACCP\P
 CHANGE (ACCOUNT # TO) 12245\5\4\2
 CHANGE (ACCOUNT # TO) 12345%
```

↑R was typed at the end of each of the first two lines.

Note: ↑R will never display a password, even if typed in a field following the actual password field.

4. (CTRL)-X, Rubout

Typing (CTRL)-X (↑X) or Rubout at any point in a command will delete the entire command. This is useful, for example, when one has made an error that can no longer be got at with ↑A or ↑W. ↑X and Rubout have exactly the same effect except that with the former TENEX types "↑X"; with the latter "XXX".

```
@RED$IRECT (INFILE) ↑X
@REN$AME (EXISTING FILE) XYZ.FOO;5$ (TO BE) XXX
@REN$AME (EXISTING FILE) ABC.FOO;1$ (TO BE) ALPHA.A;1%
```

5. Errors

Once detected, TENEX has two ways of dealing with errors. In a few cases, TENEX will type "?" after an erroneous command element and allow the user another try at making a correct entry. This happens, for example, when a supposedly existing file cannot be found, or when some other character is struck instead of a confirming carriage return. Errors

of this sort are designated "Type 2" errors in this manual.

```
@REN$AME (EXISTING FILE) XYZ. ? ABC.FOO;1
     (TO BE) ALPHA.A;1 P ? %
```

In the above, the user's first mistake was to enter a nonexistent file name - XYZ. (Note that TENEX picked this up even before he had typed the rest of the file designator.) His second mistake was to type "P" instead of the confirming carriage return that TENEX expected. In each case TENEX typed "?" and gave him a second chance.

In the more typical case TENEX will deal with errors by typing "?" or a specific error message, aborting the entire command line, and awaiting a fresh command on the next line.

```
@AV$AILABLE XYZ$ ?
@
@ST$ART
 NO PROGRAM
@
```

In each of these examples TENEX rejected the entire command. In the first case the user specified a non-existent keyword; in the second, conditions in the system prevented command execution.

For each command or group of commands in this manual a list of error messages is given. Where the message is simply "?", the error condition(s) leading up to it are spelled out. Type 2 errors are labelled as such.

In addition, there are many syntax errors such as incorrect keyword or wrong confirmation that apply to all commands. Most of these type ? and then either give the user a second chance or return to command input.

In a few unusual cases unanticipated command errors will yield obscure, systems-oriented messages from deep within the TENEX monitor. Typical of these are:

```
ILLEGAL INSTRUCTION TRAP IN EXEC
 (numbers and further message)

JSYS ERROR RETURN IN EXEC
 (further message)

EXEC SCREWUP AT PC = ...
 (more information)

ERROR WITHIN AN ERROR
 (further message)
```

These errors will abort the offending command and return to command input. If possible, they should be reported to TENEX systems personnel.

D. FORMAT CONVENTIONS IN MANUAL DESCRIPTIONS

Throughout the remainder of this manual, commands are shown as though entered with recognition input, i.e., the full command is spelled out with the minimum input characters underlined. For clarity, $ has been omitted in almost all cases, but the user should assume that altmode was struck at the end of the underlined portion of each keyword. In a few cases altmode is shown (by $) to highlight special recognition situations. Carriage return is shown by %.

The structure of each command is first spelled out fully, with all options shown. Arguments are named in lower case. Where there is a choice of keywords or arguments within a command the possibilities are all listed, separated by slash marks and sometimes enclosed in square brackets, with the default choice listed first. When null is a choice, it is indicated by "-". In the examples that follow each command description, everything is in upper case as it would be in a real typescript, and there are no extraneous brackets or slash marks.

Command descriptions:

@AVAILABLE [LINES/DEVICES]
@CHANGE (ACCOUNT # TO) account # %

Command examples:

@AVAILABLE LINES
@CHANGE (ACCOUNT # TO) 12345%

III. SYSTEM ACCESS

The commands described in this section allow the user to enter and leave TENEX, change the account that will receive charges during a session, and to release and reestablish terminal terminal control over a job. In addition, this section discusses user names and passwords, the TENEX identifiers for system users; accounts, the mechanism by which TENEX accumulates use charges; and jobs, the mechanism through which users do work with TENEX.

A. GENERAL INFORMATION

1. User Names

Each TENEX user places on file at the computer center a name, which can be any string of letters and digits of his choice, up to 39 characters long. Surnames are a typical choice. A user's name serves as his prime identifier to the TENEX system. It must be given when he logs into the system. In addition, user names assist in file sharing (see Files) and message sending between users.

2. Passwords

Because user names are generally known to the community of TENEX users, an additional mechanism is needed to safeguard privacy. To this end, each user files with the computer center a password, again a string of up to 39 letters and digits of his choice. Passwords are kept secret; generally, only he and TENEX systems personnel will know his password.

Like user name, password must be given when the user logs in to the system. To preserve password secrecy full duplex* terminals do not echo when a user types his password, and consequently nothing appears in the printed record of the

---

*Full duplex means that the terminal's keyboard and typing head are completely disconnected. Striking a key sends a character to the computer but has no effect on the typing head. For a character to be printed, it must be sent from the computer. Normally, TENEX "echoes back" the characters it receives; to preserve password privacy requires merely that this echo be suppressed.

session. With half duplex terminals**, TENEX first prints a mask of gibberish characters over which the user then types his password. Unless informed otherwise, TENEX assumes that terminals are full duplex. The user can establish his terminal as half duplex prior to LOGIN through the HALFDUPLEX command (see section IX).

A furthur complication occurs if a half duplex scope terminal is in use. These devices have a local memory which is used to refresh the display. Thus when the user types his password, the password erases the mask provided by TENEX. In order to have one's password obliterated as soon as possible, the password input should be terminated with a space. This allows the EXEC to overprint the password with the phrase "THANK YOU ... ".

3. Accounts

TENEX segregates use charges by accounts. When a user logs in, he must specify an account to receive subsequent charges. Later, he can switch to a different account through the CHANGE command. Bills from the computer center list TENEX charges by the accounts the user has specified.

Certain users (notably BBN employees) are required to specify accounts by decimal numbers of five or six digits. Other users are permitted to use account names - strings of up to 39 letters and digits. TENEX has knowledge of which category a user belongs to and adjusts the formats of the LOGIN and CHANGE commands accordingly.

4. Jobs

TENEX provides service to users through entities known as Jobs. TENEX has available a fixed pool of Jobs, each identified by a number known as the TSS Job Number. Jobs are assigned to incoming users on a first-come, first-served basis from the Job pool. If no Jobs are available when a user logs in, TENEX so informs him and aborts the LOGIN procedure.

**With half duplex terminals keyboard and typing head are physically connected; all printing characters typed on the keyboard appear on the page. Consequently, TENEX must take the action described above to safeguard passwords.

B. ACCESS COMMANDS

1. LOGIN

To gain entry to TENEX:

```
@LOGIN (USER) user name (PASSWORD) * (ACCOUNT) account name%
 [MESSAGE]
```

or

```
@LOGIN (USER) user name
 (PASSWORD)
 $%AB$%F*9(())←@+EGH>>XY????"#E=LØ)'((SST
 (ACCOUNT) account name %
 [MESSAGE]
```

The first form applies to full duplex terminals. The asterisk (not actually printed by TENEX) indicates that the user is to type his password here, but that no echo will return. The second form applies to half duplex terminals. Here TENEX types carriage return on termination of the user name, and password fields, and right after the noise word (PASSWORD). The gibberish characters are just that, the mask supplied by TENEX to disguise the user's password. Users with full duplex terminals can invoke a modification of the second LOGIN form (without mask characters) by terminating the LOGIN, user name, and/or password fields with carriage return. This places each argument on a separate line, and allows for unusually long user or account names. On half duplex scope terminals the password field should be terminated with a space so that the password will be overwritten with the phrase "THANK YOU ... ".

Both command forms shown above apply to users free to use account names. With users restricted to account numbers, TENEX types "(ACCOUNT #)" instead of "(ACCOUNT)" and accepts only a decimal number of the required form.

After a successful LOGIN, Tenex responds with a standard message giving the TSS Job Number assigned, the computer line number, and the date and time of entry, (in the form hour:min). In addition, TENEX may print a system LOGIN message, entered by the operator, that gives the latest information on system schedules, new features and the like.

TENEX will print the system message(s), if one exists, only if new entries have been made more recently than the user's last LOGIN. It is not possible to abort the system login message type out with ↑C. A user not wishing to see system messages may skip individual entries by typing RUBOUT (DEL

or DELETE on some terminals) during its output.

If private mail exists for the user, TENEX will type "YOU HAVE A MESSAGE" immediately following the LOGIN information and the system messages. Private mail may be read by using subsystems such as READMAIL. READMAIL may also be used to browse through old system LOGIN messages.

One user may leave mail for another by using the SNDMSG subsystem. SNDMSG allows sending messages to users of distant ARPANET compters as well as local users.

In addition to the above mentioned information, TENEX will inform the user if he has exceeded his disk space allocation.

Examples:

```
@LOGIN (USER) SMITH (PASSWORD) (ACCOUNT #) 11451%
JOB 6 ON TTY25  13-APR-73 12:34
TENEX WILL GO DOWN FRI 13APR73 22ØØ TIL SAT 14APR73 Ø2ØØ
YOU HAVE A MESSAGE
```

Here, a user restricted to account numbers entered on a full duplex terminal at 12:34 p.m., on April 13, 1973, assigned Job 6 and line 25, and was notified of a pending system shutdown and of the existence of unseen mail.

```
@LOGIN (USER) JONES
(PASSWORD)
DJDIOURU12343AJDS9Z,CSDHD123845JJD73F;SKØ-:23$$%%$"!'9
(ACCOUNT) ADMINISTRATIVE%
JOB 2 ON TTY13 13-APR-73  9:Ø3
JONES OVER ALLOCATION BY 47 PAGES
```

This user entered on a half duplex terminal at 9:Ø3 in the morning. He specified an account name and was assigned Job 2.

```
@LOGIN (USER) EXCEPTIONALLYLONGUSERNAME%
(PASSWORD)%
(ACCOUNT) UNUSALLYCOMPLEXACCOUNTDESIGNATOR%
JOB 4 ON TTY17  13-APR-73  1Ø56
```

Because of their length, this user terminated the arguments with carriage return.

Errors: YOU ARE ALREADY LOGGED IN
"?" after user name if no such user
"?" after password if incorrect
"?" after account if must be numeric but isn't

Notes:

LOGIN is usually the first command a user will give after gaining TENEX's initial attention with ↑C (see II-A-1). Generally the facilities and commands of TENEX are inaccessible until a user has logged in. There are three exceptions. Certain system queries may be used to obtain information about the system prior to LOGIN. These are noted with their individual descriptions in section VIII. The terminal characteristics and tab stop commands described in section IX can be used at any time to communicate the nature of the user's terminal to TENEX. The ATTACH command described below can be used prior to LOGIN to regain control over a DETACHED Job.

AutoLogout: If, after typing an initial ↑C to TENEX the user fails to LOGIN within a reasonable time, TENEX will drop him from further attention through the Autologout function. Upon completion TENEX will type a message similar to the standard LOGOUT message below, giving the elapsed and CPU times consumed. After an Autologout the user can once again initiate discourse with TENEX by typing ↑C.

2. LOGOUT

To leave the system:

@LOGOUT%

LOGOUT closes any currently open files, clears the user's Job, and returns it to the available Job pool. After this has been done, TENEX types a message indicating the Job and user logged out, the current account and computer line, and the time of LOGOUT. Just prior to this point, the user's disk space allocation is checked against that which he has in actual use, and an appropriate message typed if he is over his assigned limit. (Increases in disk allocation may be negotiatiated with the Computer Center Operations Manager.) Following this, TENEX types the processor and connected times, in the format hours:min:sec.

If connection was over telephone lines, TENEX hangs up its end of the connection. Use charges cease at the moment the user types the terminating carriage return.

Example:

```
@LOGOUT%
KILLED JOB 6, USER SMITH, ACCT 11451, TTY 25, AT 14:56:34
  USED Ø:Ø:4 IN Ø:21:5
```

TENEX allows LOGOUT to be stopped with a ↑C until the very instant that it takes effect. TENEX will respond with "NOT LOGGED OFF" if this is the case. Note however, that an EXPUNGE operation will have already been performed on the connected directory.

If a user has several Jobs logged in, he may LOGOUT any of them without ATTACHing them. This is done by supplying the Job number as an argument to the LOGOUT command.

@LOGO 14%

The above command would logout Job 14 if it is owned by the same person.

Errors: NOT LOGGED OFF
IF YOU WANT TO LOGOUT THIS JOB, USE LOGOUT
THAT JOB DOES NOT EXIST
NOT LEGAL IN INFERIOR EXEC

3. CHANGE

To change accounts for subsequent computer use charges:

@CHANGE (ACCOUNT TO) account name%

or

@CHANGE (ACCOUNT # TO) account number%

This command allows the user to switch computer use charges from one account to another during the course of a working session. The format depends on whether the user can use account names or is limited to numbers. Use charges begin accruing to the new account on termination of the command, and continue to do so until LOGOUT or a subsequent CHANGE command.

Examples:

@CHANGE (ACCOUNT TO) OVERHEAD%

@CHANGE (ACCOUNT # TO) 12345%

Errors: "?" if account must be numeric but isn't.

4. DETACH

Occasionally the user may wish to initiate a TENEX Job, start it running, and then disconnect his terminal from it. This would save unnecessary telephone charges, for example,

when a lengthy processing operation was started from a distant terminal, or permit a user to use his terminal for other purposes while his Job ran to completion. The TENEX DETACH command makes this possible; its format is:

@DETACH (INFILE) [-/existing file designator] (OUTFILE) [-/file designator] (AND) [-/START/REENTER/CONTINUE]

The INFILE and OUTFILE arguments specify files that will take the place of the user's terminal once the DETACH operation is complete. If the job requests terminal input it will come instead from INFILE; any terminal output will be sent to OUTFILE. See IV-A for the format for typing file designators. The START, REENTER, and CONTINUE keywords permit starting a user program or subsystem that may have been loaded into core prior to DETACH. Their action is identical to the START, REENTER, and CONTINUE commands described in VII-A-6,7,8.

If INFILE and OUTFILE are given, but no program is started up, then TENEX will seek further Executive commands from INFILE. If a program is started, then any terminal input or output it generates will affect INFILE and OUTFILE. If a program is started in the absence of INFILE and OUTFILE all will proceed normally until a request for terminal input or output occurs, at which point the job will go into a dormant state. If no arguments are specified the job will go dormant immediately, since TENEX will immediately request, but find no source for, Executive commands.

In the 4/13/73 version of TENEX, the INFILE and OUTFILE options are unavailable; these fields must be nulled. This limits the uses of DETACH to the last two cases described above. However, this can still be useful when one wishes to start a lengthy processing operation known not to request terminal input or output, or to preserve temporarily an elaborate job structure.

Examples: The following illustrate the two options currently available with DETACH.

@DET%

@DETACH (INFILE) $- (OUTFILE) $- (AND) START%

In the first example the user's job will remain dormant until subsequent reconnection through the ATTACH command below. In the second example any processing triggered by the START option will take place, after which the job will become dormant.

Errors: error in file name (" ? ")
error in opening file
errors in START/REENTER/CONTINUE: see same.

5. ATTACH

To regain control over a DETACHed job:

@ATTACH (USER) user name (PASSWORD) * (TSS JOB #)
job number % or $

The asterisk indicates that the password is not echoed on full duplex terminals. With half duplex terminals TENEX provides a modified format similar to LOGIN. The TSS Job Number specifies the Job to be ATTACHed to; it is the Job Number that was assigned when that Job was first logged in. If the Job Number field is defaulted, TENEX will select the DETACHed job belonging to this user if exactly one such job exists.

ATTACH can be given without logging in, since the user is re-establishing control over an already logged in Job. A user may successfully ATTACH any job for which he knows the appropriate password, even if that Job is not currently detached. (As with all commands which require a password, the password field should be terminated with a space if a half duplex scope is in use.) If the user is logged in at the time of an ATTACH command, his current job is automatically DETACHed, and TENEX types the message "DETACHING JOB n".

If the ATTACHed-to job was running at the time of ATTACH, it continues doing so. If dormant due to lack of terminal input, it comes back to life.

Example:

@ATTACH (USER) JONES (PASSWORD) (TSS JOB #) 2%

Errors: "?" after user if no such user
THAT'S A FILES-ONLY DIRECTORY NAME
"?" after password if incorrect and
current job is not logged in
JOB n NOT LOGGED IN
JOB n NOT LOGGED IN UNDER name
NO DETACHED JOB LOGGED IN UNDER name
TSS JOB # REQUIRED - name HAS
MORE THAN ONE DETACHED JOB
JOB NUMBER MUST BE BETWEEN 1 AND 39
(Note: 39 is a system dependent parameter)
INCORRECT PASSWORD (if logged in)

IV. TENEX FILE SYSTEM

The TENEX file system provides two facilities. First, TENEX files provide a means for storing and retrieving information over and above the use of core memory during a working session. TENEX files can be kept on disk, DECtape, ordinary magnetic tape, or paper tape. Files stored on these media can contain any information that TENEX is capable of handling.

In addition to this conventional use of files as a storage medium, TENEX file commands give the user access to input/output devices not usually associated with files. For example, the line printer and the user's terminal can both be treated as files, the one capable of receiving information, the other capable of receiving or supplying it. This treatment of input/output devices as files unifies and simplifies the moving and storing of information in TENEX. Often, a single TENEX file command can perform input and output as well as file manipulation, depending on the arguments given it.

Executive commands discussed in this section, together with commands in the various subsystems, give the user control over his files. They allow him to create and destroy files, specify where in the system they are to be kept, make copies of them, and cause them to be output.

A. TENEX FILE DESIGNATORS

In order to bring an executive command to bear on a file, one must identify the file in question by giving its TENEX file designator. Depending on circumstances, a TENEX file designator may contain up to five* file descriptors that specify the file's name, the device it is to be sent to or taken from, and certain other characteristics. With all five descriptors included, a TENEX file designator has the following form:

device:<directory name>file name.extension;version number

Underlined characters in the above are typed literally. Other characters define the five descriptors that make up a file designator. The paragraphs that follow explain the meaning and syntax of the file descriptors. In practice, of course, file designators appear as arguments to TENEX

---

*Additional optional file descriptors, applicable only to disk files are covered in IV-C.

executive commands. This means that the entire designator, taken together should be terminated with space, alt mode, or carriage return, depending on its position in a TENEX command and the TENEX rules for abbreviation and file designator recognition.

1. Device

TENEX keeps separate track of files stored on separate devices. Consequently, in dealing with a TENEX file, one must specify the device it is to be taken from or sent to. TENEX devices are designated by device codes (usually three or four letter abbreviations), and to specify a device, one must type its code, followed by a terminating colon. With devices such as DECtapes, magnetic tape units, and terminals, TENEX may have available more than one unit. In this situation, a unit number must be appended to the device code, e.g., MTA3:, TTY27:. However, the user's terminal can always be referred to as just TTY:.

TENEX cannot perform recognition on device codes, and when a code is specified it must be typed in full. However, if the user omits the device altogether from a TENEX file designator (by starting with directory or file name), TENEX will assume (but not type) the default value disk (DSK:).

In addition to its ordinary storage and input-output devices, TENEX has a unique ecological stabilizer, NIL:, which has no physical embodiment but serves as a drain capable of absorbing limitless amounts of unwanted information. It also functions as an inexhaustible source of zeros. Files written onto NIL: simply disappear without trace, never to be seen again. No obsolescing data is left behind to pollute TENEX or the external environment. NIL: can also be viewed as a write only memory.

TENEX devices and their codes are listed below.

DEVICE	CODE
Disk	DSK:
Magnetic tape units	MTAØ:-MTA3:
DECtape units	DTAØ:-DTA3:
User's terminal	TTY:
Specific terminal lines	TTYØ:-TTY12Ø:
Line printer	LPT:
Paper tape reader	PTR:
Paper tape punch	PTP:
Plotter	PLT:
Private disk packs	DPKØ:-DPK3:
Digital to analog converter	DAC:
Analog to digital converter	ADC:
Disposal	NIL:

When accessing disk, private disk packs, or DECtape files, additional descriptors are needed to identify the particular file intended. The remaining devices, however, serve as sources or sinks for unnamed information, and with these devices the device code alone constitutes a complete file designator. For example

PTR:

serves as a complete designator for any file to be input via the paper tape reader, and

LPT:

completely specifies the line printer as a destination.

2. Directory name

Files stored on the TENEX disk are grouped into file directories, each designated by a directory name. Each registered TENEX user has one "primary" directory with a directory name identical to his user name; he may also have additional file directories bearing other directory names. A TENEX user can access the files in any of his directories, and if authorized*, files in the directories of other users as well.

At any moment in time, a TENEX user is "connected" to a given file directory. When he first logs into the system, connection is automatically established to his primary directory. At any time the user can establish connection to

---

*Access to disk files is controlled through an optional descriptor that specifies protection status, as discussed in Section IV-C-2 below.

another directory, his own or someone else's, through the CONNECT command described in IV-E-13 below. In accessing a file in the directory to which he is connected, the user can omit directory name from the file designator. TENEX will assume (but not print) the connected directory name as a default value. To access files in other directories, the user has two options. He can shift connection to the desired directory (through CONNECT) and then proceed as above. Alternatively, he can name the directory in question as part of the file designator. In this case, he accesses the desired file, but remains connected to the original directory for future file reference.

Directory names begin and end with opening and closing angle brackets:

```
<SMITH>
<JONES>
```

TENEX will perform recognition on a directory name if a partially typed name is terminated by altmode or ↑F.* In recognizing the directory name, of course, TENEX checks a list of all names currently registered. Consequently the number of characters needed for recognition of a particular name will depend on the TENEX installation, and may vary over time.

Files stored on DECtapes or magnetic tapes are also kept track of through directories, but each DECtape or magnetic tape has just one unnamed directory that lists all the files it contains. Consequently, directory name is meaningless, and need not be supplied in designating files stored on DECtape or magnetic tape.

3. File Name

Every disk, DECtape, or magnetic tape file carries a name given to it at the time of its creation. A file's name serves as the primary handle for gaining access to it. A TENEX file name can be any combination of letters, digits, and some punctuation marks (see JSYS Manual) up to 39 characters long (6 characters for DECtape). The following are legitimate TENEX file names:

```
1
A
VERYLONGFILENAME
```

---

*Section IV-B-3 explains file designator recognition in detail.

As with directory names, TENEX will recognize file names when they are terminated with altmode or ↑F. In recognizing file names, TENEX looks only at the directory in which the file resides. Consequently, the user has control over name recognition in his own directory(s) and, by judicious use of initial characters can construct file names that require little input for recognition. For example, by using a different first character each time, the user could construct up to 36 file names, each recognizable after one character of input.

4. Extension

Frequently it is desirable to create several files with the same file name, containing perhaps related information or different transformations of the same information. TENEX file extensions make this possible with disk, DECtape, and magnetic tape files by modifying the file names they follow, and distinguishing between separate files with a common name. For example:

```
ALPHA.F4
ALPHA.REL
ALPHA.SAV
```

might designate symbolic, relocatable binary, and core image versions of the program contained in file ALPHA.

TENEX file extensions consist of a period, followed by any combination of up to 39 letters and digits (3 with DECtape). Although TENEX regards all disk, magnetic, and DECtape files as having extensions, the user can specify a null extension (one having no characters) by typing just the initial period.

```
.
.A
.EXTENSIVEXTENSION
```

TENEX will recognize file extensions, limiting its search to the extensions associated with the current file name. If a null extension is present, TENEX will recognize it when the user types no extension characters at all, i.e., if he types altmode or ↑F at the beginning of the extension field.

The user can freely invent file extensions and use them for any desired purpose. However, for convenience, TENEX subsystems recognize a standard collection of file extensions, each of which specifies a certain type of data. In many cases subsystems will select the appropriate extension when an input file name is given, and automatically supply the standard extension when creating

output files. Some of the standard extensions are listed below.

EXTENSION	MEANING	RELEVANT SUBSYSTEMS
.MAC	a MACRO-1Ø source program	MACRO-1Ø
.BAS	a BASIC program	BASIC
.F4	a FORTRAN IV source program	FORTRAN IV
.REL	a relocatable object program	FORTRAN IV MACRO-1Ø, LOADER
.SAV	an executable object program (core image)	FORTRAN IV MACRO-1Ø, DDT

5. Version Number

TENEX version numbers carry the idea of file extensions one step further and put it to use for a different purpose. Version numbers are available only with files stored on the disk; they allow one to create multiple disk files with the same combination of file name and extension. The intent of version numbers is to keep track of successive, modified versions of a single file. For example,

```
ALPHA.F4;1
ALPHA.F4;2
ALPHA.F4;3
```

might indicate three successive versions of a FORTRAN IV source program, ALPHA.F4, each differing somewhat from the one before.

In the above, ;1, ;2, and ;3 are TENEX file version numbers. TENEX version numbers consist of a semicolon followed by a decimal integer between 1 and 262141. Within this range the user can freely specify version numbers when creating TENEX files. TENEX does not "recognize" version numbers in the sense used in this manual. However, if the user omits version number from a disk file designator (by typing space or altmode after the file extension), TENEX will pick a default value according to the following rules.

1. If this is the creation of a new file (as indicated by new name.extension;), TENEX assigns 1.
2. If this file is to be read from storage, TENEX

selects the highest-numbered version.
3. If the file is to be written onto storage, TENEX assigns a number one greater than the highest existing number.
4. If the file is to be deleted, TENEX picks the lowest numbered version.

In reading from storage, TENEX complains if a user-specified version is not found. In writing, if the indicated version is already present, the new information replaces the information already in the file. If the user enters Ø as version number, TENEX will supply (but not type) the highest existing number. This is useful, for example, when one has forgotten the current highest version, but wishes to avoid creating successive versions under rule 3 above. Version number "-2" is a shorthand denoting the lowest version number. Thus, the command:

@DELETE *.*;-2%

will delete the lowest numbered version of every file in the connected directory. Version number "-1" is another way of specifying the next higher version option for version numbers (see IV-C-2).

Although version numbers can exist anywhere in the allowable range, each user is restricted as to how many versions he can keep of any one file. This restriction is controlled by an allotment number assigned to the user when he registers with the system; it can be changed, if desired, by the system operators. The restriction works as follows. If the user exceeds his allotment by creating a new file version, then the earliest existing version (in terms of date and time last written) is automatically deleted by TENEX. For example, if the user's allotment were 3 and he wrote versions ;27, ;16, and ;43 of a given file on successive days, then writing a new version would cause version 27 to disappear. Note that the restriction has nothing to do with the order of version numbers, only the order in which files were written (although frequently the two will coincide). Note: Many installations do not implement this feature.

B. TYPING FILE DESIGNATORS

The five part structure of TENEX file designators reflects the convenience and uniformity of the TENEX file system, most of which would be lost if the user had to enter all five descriptors every time he accessed a file. TENEX avoids this pitfall by three mechanisms alluded to in the foregoing discussion. First, some descriptors are relevant only to certain files and need not be supplied with others.

Second, in other situations, TENEX will supply default values for relevant descriptors omitted by the user. Third, TENEX will apply recognition to some of the file descriptors.

1. Relevant Descriptors

All five descriptors are relevant to disk files. With other files, only some of the descriptors apply. Thus, as shown below, many TENEX files can be specified by abbreviated versions of the full file designator.

DEVICE NAME	APPLICABLE FILE DESCRIPTORS
DSK: DPKØ:-DPK3:	device name:<directory name> file name.extension;version
MTAØ:-MTA3: DTAØ:-DTA3:	device name:file name.extension
TTY: TTYØ:-TTY12Ø: PTR: PTP: LPT: PLT: ADC: DAC: NIL:	device name:

2. Default Values

TENEX will supply default values for device name, directory name, and version number. Device name defaults to DSK: . Directory name and version number are relevant only to disk files. Thus the general default capability permits abbreviated versions of disk files as summarized below.

DESCRIPTOR	DEFAULT VALUE
device name:	DSK:
directory name	currently connected directory
file name.	-----
extension;	-----
version	see detailed rules (IV-A-5)

Thus, in many situations, the user need only specify file name and extension to designate a disk file. In addition to the general default capability summarized above, with certain file commands TENEX will default file name and/or extension. These situations are described as they arise.

3. Recognition

As discussed separately in preceding sections, TENEX will recognize directory name, file name and extension. Unlike the situation with command recognition, the user has two ways of triggering recognition with file descriptors - altmode or ↑F. Otherwise, descriptor recognition is similar to command recognition: the user must type sufficient characters to determine a match. With insufficient characters TENEX rings the terminal bell and awaits further input. With unrecognizable input, TENEX types a question mark and allows the user to try again.

If the input is recognized, further action depends on whether the user terminated with ↑F or altmode. In either case, TENEX types the remainder of the descriptor, including the character >, ., or ; as appropriate. With ↑F, TENEX now pauses to allow the user to enter the next descriptor. With altmode, TENEX attempts to determine all remaining descriptors, applying recognition to file name and extension, and the default rules described previously to version number. TENEX types each descriptor successfully determined; it pauses and rings the terminal bell on encountering any descriptor it can't determine.

Unexpected results can occur unless the user keeps in mind that TENEX will recognize a null extension in the absence of specific input. For example, if one triggers file name recognition by altmode, TENEX will select a null extension associated with that name even though there may be other extensions. TENEX will also select a file name or extension without specific input if it is uniquely specified. For

example, if a given directory contains just one file, then terminating the directory name with altmode will cause TENEX to select the file in question.

4. Examples

The net effect of the rules and conventions outlined above is to reduce the amount of typing required in specifying TENEX files. Assume that DECtape 1 contains:

```
ALPHA.ABC
BETA.XYZ
```

and that the disk contains:

```
<SMITH>ALPHA.ABC;1
<JONES>ALPHA.ABC;1
<JONES>BETA.;1
<JONES>BETA.XYZ;1
<JONES>BETA.XYZ;2
```

then with user JONES typing, the following would be complete TENEX file designators.

1) LPT:
2) NIL:
3) DTA1:A$LPHA.ABC
4) <S$MITH>ALPHA.ABC;1
5) A$LPHA.ABC;1
6) B$ETA.;1
7) B↑FETA.X$YZ;2
8) B↑FETA.X↑FYZ;1

Examples 1 and 2 are device names that form complete file designators. Example 3 illustrates the three-part structure of a DECtape file designator, and also shows name recognition triggered by altmode. Note that TENEX selected the only extension present. In example 4, device name defaulted to disk. Also, user SMITH has but one file in his directory, so terminating his name with altmode caused automatic selection of the last three descriptors.

In the remaining examples device name and directory name have both defaulted, the latter taking on the value JONES. In example 5, the entire file was recognizable after the first character of its name. Example 6 shows a case where altmode triggered selection of the null extension, even though others were present. In example 7, ↑F was used to terminate BETA, so that extension XYZ could be selected. This example assumes the file was to be read from the disk; version 2 was selected according to the default rules described previously. In example 8 the extension was

terminated with ↑F so that version 1 of the same file could be selected by hand.

C. MULTIPLE FILE DESIGNATORS

Certain TENEX commands can address several files at one time. For example, one can APPEND a series of files onto an existing file, or DELETE several files in one operation. In cases where more than one file is legal, there are two ways to enter multiple file designators.

1. File Designator Lists

One can list a series of file designators separated by commas:

DTA1:ALPHA.ABC;1,BETA.;1,<SMITH>ALPHA.ABC;1

After the first designator, name and extension default to those in the preceding designator (but other descriptors do not). With some commands one can terminate a file designator with space or altmode before typing the separating comma; with others, the comma itself must terminate all designators but the last. In the latter case, only ↑F can be used to trigger recognition or defaulting, since altmode might prematurely terminate the entire list. Assuming a command of the second kind, the following illustrates the defaulting of file name and extension:

B↑FETA.;1,↑FBETA.XYZ;1,↑FBETA.↑FXYZ;2

The first ↑F triggered recognition of name BETA. The second and third defaulted name to its previous value, while the fourth defaulted extension to XYZ.

2. The * Feature

As a second way of designating multiple files, one can enter "*" in place of directory name, file name, extension, or version number. When used this way, * means "all existing values of". Assuming the disk files of IV-B-4, and with Jones typing:

Example	Files Accessed
*.XYZ	All files in connected directory with extension .XYZ, that is:

	BETA.XYZ;1 BETA.XYZ;2
BETA.*;1	All files in connected directory named BETA and with version 1: BETA.;1 BETA.XYZ;1
<*>ALPHA.ABC	All ALPHA.ABC's in any directory: <SMITH>ALPHA.ABC;1 <JONES>ALPHA.ABC;1
<SMITH>*.*;*	All files in directory <SMITH>: <SMITH> ALPHA.ABC;1.

When the version is omitted (as in the first and third examples above), which version is selected depends on the TENEX command. Some commands will select all versions, others the highest existing version.

When multiple file designators are legal, they can be specified by a list of files, by the * feature, or by a mixture of the two. For example, the following would specify all Jones's .XYZ files together with ALPHA.ABC;1:

.XYZ;,A$LPHA.ABC;1

When processing multiple file designators, TENEX types the separate designators for the files actually accessed. This is particularly useful in verifying TENEX's interpretation of any *'s that may have been given.

D. OPTIONAL DISK FILE DESCRIPTORS

In addition to the five principal file descriptors which serve mainly to identify and organize files, there are three optional descriptors, available only with disk files that can be used to establish file properties. Through these descriptors, one can establish the protection status of a file and the account to be charged for storing it, or one can declare the file to be temporary.

These descriptors need not be supplied in designating a file; TENEX will always assume default values. However with optional descriptors included, the complete file designator for a TENEX disk file takes on the following, rather

horrendous form.

DSK:<directory>name_file name.extension;version number
;T;Pprotection status;Aaccount

The format for the three optional descriptors is similar: each begins with a semicolon followed by the letter T, P, or A to indicate which descriptor is intended. Because they are identified by their leading characters, these descriptors can be entered in any order relative to each other (all must follow the five main descriptors). However, as discussed below, if the descriptor ;T is entered, then neither protection status nor account can be specified. TENEX performs no recognition on these descriptors; if entered at all they must be entered completely. The meaning of these descriptors is spelled out below.

1. Temporary Files

On creating a new file, one can declare it to be temporary by entering ;T after the main descriptors. A temporary file will be deleted automatically by TENEX when the user logs out. Temporary disk files are useful for such things as testing programs, or collecting data for listing on the line printer or later transfer into permanent files. Any disk file can be declared temporary, but a temporary file can be given neither special protection nor account status; TENEX will automatically supply the default values described below for these descriptors. If the ;T descriptor is omitted from a file designator, TENEX will treat it as an ordinary, permanent file.

2. File Protection

In general, one can apply any TENEX file command to any disk file, no matter who it belongs to as long as one can specify the descriptors that make up its file designator. Furthermore, even when one's knowledge of these descriptors is incomplete, TENEX can help fill in the gaps through its recognition and default value features. Thus, by its nature, TENEX provides a very general file sharing capability. This leads to an equally general need for some way of protecting files from unauthorized access.

To this end, TENEX provides a file protection mechanism, exactly balancing its file sharing capability. File protection is controlled by a six digit octal number assigned to each file at the time of its creation. This number breaks down into three fields, identical in format, each of which can be regarded as containing two octal digits or six binary bits. The bits in any one of these fields

control the following aspects of file protection.

FILE		PROTECTION			FIELD	
BØ	B1	B2	B3	B4	B5	

Bit	Protection Aspect Controlled
BØ	Read contents of file
B1	Write onto file
B2	Execute program stored in file
B3	Append to file
B4	Access per page table
B5	Not used

Setting a bit 1 permits the indicated action; setting it Ø denies the action. Read allows information to be extracted from a file. Write permits new information to be written onto a file, replacing part or all of the original contents. Execute allows a file that has been read into core memory to be executed as a program. Any file can be made executable in this sense; what happens when one tries to start execution, of course, depends on what the file actually contained. Append allows new information to be added to the end of a file. "Per page table" access means that read, write, and execute access will be specified by the file itself for each individual page. The remaining bit is unused and can take on either value with no effect.

Taking these six bits as a two digit octal number, some common values are: 77, which permits full access; 52, which protects a file from modification, but permits other functions; and ØØ which denies everything.

As mentioned above, a full, six digit file protection number contains three of these protection fields, arranged as follows:

FILE	PROTECTION	NUMBER
self	group	others

Working from left to right the three fields control access to the file by successively larger groups of users. Self protection governs one's own access to a file. Limiting one's own access permits one to protect valuable information from inadvertent destruction, or to protect a file from modification by a possibly faulty program.

3. Directory protection

In addition to allowing detailed specification of the access to file contents, TENEX also allows the contents of directories to be protected in a similar way. The format of the directory protection word is composed of three 6-bit fields, one field for each of "self", "group", and "others" similar to the file protection word. The bits have meaning as follows:

Bit	Protection Aspect Controlled
BØ	If off, completely prevents use of the directory in any way
B1	Files may be opened subject to file protection
B2	Owner-like functions may be performed (including CONNECT) without password
B3	Files may be added to the directory
B4	Not used
B5	Not used

The directory protection word may be changed by contacting the TENEX Operations Manager.

4. Groups

TENEX provides a mechanism whereby users can form groups for the purpose of file sharing. This is useful, for example, where several users are collaborating on a common programming job and wish to share the files they are creating. The file group mechanism works as follows. Each group is assigned a number. For each user, TENEX records a series of these numbers indicating what groups he belongs to. Likewise, for each file directory, TENEX records numbers indicating which groups have access to the files it contains. Thus a given user can belong to a number of groups. He can make each of his directories available to one or more groups, which may or may not coincide with the groups he belongs to. The assignment of group numbers to users and to file directories is done by the TENEX system operators.

The second field in a file protection number governs the powers granted when the file is accessed through the group mechanism. If a group member designates a file in a directory available to his group but belonging to another user, it is this protection field that controls the kind of access granted him.

The third protection field governs the access granted to users approaching the file from outside the group mechanism, i.e., all "other" users. In a typical situation, the three protection fields might specify successively more stringent protection. For example a user might grant full access to himself, read, execute, and append access to group members, and append only access to others, yielding ;P7754Ø4 as protection descriptor.

If a user omits a protection number when creating a disk file, TENEX will assign a default value, presently 777754. This grants full access to the user and group members, read, execute and append access to others. The Default File Protecion word may be changed by contacting the TENEX Operations Manager.

5. Account

On creating a disk file, the user can specify the account to be charged for its subsequent storage. Depending on his status with respect to account information (as described under System Access, section III-A-3) the user will be required to specify an account number of five or six decimal digits, or permitted to enter an account name of up to 39 letters and digits. For example

```
;A11451
;AOVERHEAD
```

would be legitimate account descriptors for the two classes of users. If the user fails to specify an account descriptor, TENEX will assign the account name or number currently in force (as determined by the LOGIN (III-B-1) or CHANGE (III-B-3) commands). The user can change the account to which a file is charged through the CHANGE command (IV-F-6).

6. Examples

In the following, new disk files are created with the optional descriptors applied.*

```
ALPHA.ABC;2;T
ALPHA.ABC;3;A12345;P7752Ø2
```

---

*Depending on how the files were created these designators might require less typing than the amount shown. For example, the COPY command (IV-E-3) will supply default values for extension and version, and sometimes for file name.

Here a temporary version of ALPHA.ABC was first created. It will disappear when the user logs out. Next, an ordinary (permanent) version was created, to be charged to account 12345, and carrying protection status 775202.

E. FILE DESIGNATOR ERRORS

Some or all of the following errors apply to every command containing a file designator argument:

1) "?" if file not found and existing file required (type 2).
2) NO JFN'S AVAILABLE: YOU MUST CLOSE SOME FILES FIRST
3) JSB FULL: TRY CLOSING SOME FILES THEN REPEATING COMMAND
4) DIRECTORY FULL: CAN'T CREATE NEW FILES UNTIL YOU DELETE SOME FILES AND "EXPUNGE (DELETED FILES)"
5) NEW FILE NAME REQUIRED
6) device: IS NOT MOUNTED
7) device: IS ASSIGNED TO JOB n
8) READ PROTECT VIOLATION FOR FILE name
9) WRITE PROTECT VIOLATION FOR FILE name
10) FILE name BUSY
11) BAD USE OF *
12) NO FILES IN THAT DIRECTORY

These error messages have the following meaning.

1) TENEX expected the user to designate an existing file, but can find no file matching the designator actually typed.

2) Before a file can be manipulated (read, written, appended to, etc.) it must be opened. Each time a file is opened, it is assigned a Job File Number, and each user job has available a limited pool of these numbers. After manipulation, an open file is normally closed, and its JFN returned to the free pool. Files can be opened by user programs, and various other mechanisms within TENEX, and can be kept open as long as necessary. In addition, many TENEX File commands must open the file(s) designated by the user as a first step in performing the action indicated. (When the action is complete the file is automatically closed.) When a File command must open a file, but finds no JFN available, this message is typed. The user can list the assigned JFN's with FILSTAT (VIII-8) and then free up one or more of them through SHUT or JFNCLOSE (IV-F-11,12).

3) For each job, TENEX maintains a Job Storage Block (JSB) to contain miscellaneous status information necessary for running the system. Among other things, the JSB contains data on each open file. When a TENEX File command attempts

to open a file but finds the JSB full, this message is typed. Closing one or more files by SHUT or JFNCLOSE will probably free up enough space in the JSB to open a new file.

4) Each disk file directory has a limit, established by the system operators, on the number of files it can hold. When this limit is reached, attempts to create new files will yield the error message shown. Removing files from the directory through DELETE followed by EXPUNGE (IV-F-7,9) will free up space for new files.

5) The user designated an existing file in a context where a new file was required.

6,7) Prior to accessing a DECtape, private disk pack, magnetic tape unit, or the paper tape reader, the user must reserve the device in question for his exclusive use through one of the device handling commands described in detail in Chapter V. An attempt to access one of these devices without first reserving it will yield one of the error messages shown. Message 6 will also result if one tries to access a DECtape drive with no tape, or the line printer when it is off line.

8,9) The attempted access violated the read or write protection status (IV-C-2) of the specified file.

1Ø) The TENEX command has attempted to open a file that is already open. The file may have been opened by another job, or due to an error or program interruption it may be held open by the user's job.

11) * was used in a context where a file group (IV-C-2) cannot be designated. This message also occurs if the user attempts to use * as an ordinary character in a file descriptor.

12) * was given for name, extension, or version with a directory that contains no files.

F. FILE COMMANDS

The TENEX executive commands described below list the contents of file directories, move files within the system and to and from input/output devices, change file names and file properties, and destroy and revive files. Additional commands, described in Section VII, create files from information contained in TENEX core memory and load these files back into core.

1. DIRECTORY

To obtain information about files:

@DIRECTORY [device:][<directory name>]
[file name.extension;version] % or ,%

The three arguments are optional. Omitting them altogether leads to a tabulation of the disk directory to which the user is connected. By the selective use of the arguments DIRECTORY can be made to tabulate a disk directory other than the connected one, the directory for a device other than the disk, or information about a particular file. For example:

@DIRECTORY%	all files in the connected directory
@DIRECTORY <SMITH>%	all files in directory "SMITH"
@DIRECTORY DTA1:%	all files on DECtape 1
@DIRECTORY <$SMITH>ALPHA.FOO;1%	just the selected file
@DIR <*>FOO.*;*	all files on the DSK: with name FOO

DIRECTORY can tabulate a variety of information about files in several different formats. The format and the information listed are controlled by a series of optional sub-commands that modify the action of DIRECTORY. In the absence of subcommands, DIRECTORY types, on a separate line for each file, the name, extension, and version number of each file selected by the arguments given. Files are listed alphabetically. Where two or more files have the same name, only extension and version number are typed for files after the first. Where there are several versions of a file, version numbers are typed on one line, separated by commas and following the file name and extension. In the case of temporary files, TENEX types ;T after the version number. To obtain a listing of this sort, terminate DIRECTORY with carriage return. In the following examples, user Jones is connected to his primary directory.

```
@DIRECTORY%

 ALPHA.ABC;1
 BETA.  ,1
    .XYZ;2,1
@
```

To modify DIRECTORY with sub-commands, terminate with comma-carriage return. TENEX will type "@@" and await subcommand input. Certain subcommands allow additional keyword fields to specify alternatives. When more than one alternative is available, the allowable keywords are shown separated by slash marks, with the default value given first. Subcommands are terminated with carriage return. TENEX continues accepting sub-commands until the user types "BEGIN", or a carriage return with nothing before it, at which point the requested action begins. Subcommands are listed below by function.

a) Additional data to print: The following sub-commands cause TENEX to print additional information for each file selected (over and above the minimal name.extension;version number).

@@ACCOUNT%
TENEX prints the name or number of the account that will receive storage charges for this file. (See Section III)

@AUTHOR%
The name of the user who last wrote the file.

@@PROTECTION%
The protection status of the file.

@@SIZE%
TENEX prints size of file in "pages". A TENEX page contains 512 PDP-1Ø words.

@@LENGTH (IN BYTES)%
TENEX prints size of file in "bytes". A byte is a portion of a full computer word; its length is measured in bits. The byte length for a given file will depend on the information stored. Text files, for example, contain 7 bit bytes. TENEX types the byte length applicable in parentheses after the length.

@@DATES (OF) WRITE/READ/CREATION%
TENEX prints the date on which the file was most recently written or read, or its creation date, depending on the keyword selected. To obtain the most recent date for more than one of the possible operations, one can give repeated DATES commands, each time specifying a separate keyword.

@@TIMES (AND DATES OF) WRITE/READ/CREATION%
TENEX prints the most recent time and date of the operation. As above, the command can be repeated to document more than one operation.

@@VERBOSE%
TENEX prints the standard file information (name, extension, version numbers, ;T if temporary) plus account, protection, size in pages, dates of last write and read, and the author. This information just fits on one terminal line.

@@EVERYTHING%
TENEX prints everything printable about each file. This includes the standard file information plus everything that can be selected by the above subcommands.

@@TEN5Ø%
An obsolete directory listing format.

b) Printing Format. In listing file information, TENEX normally single spaces, columnizes the various fields of information, and provides a tabular heading if information is requested beyond account and protection. The following commands alter the normal format.

@@NO (HEADING)%
Suppresses heading, if any.

@@DOUBLESPACE%
Output is double spaced.

@@SEPARATE (LINES FOR EACH VERSION)%
Forces a separate listing for each version of a file. This will happen automatically if information printed by one of the subcommands differs by version.

@@CRAM%
Suppresses columnization. The separate items of information for each file follow one another regardless of length. This produces faster but less intelligible directory listings.

c) Order of Printing

@@ALPHABETIC%
The directory listing will have files in alphabetic order.

@@CHRONOLOGICAL (BY) READ/WRITE/CREATION% Order of printing will be determined from the indicated date.

@@REVERSE% This command may be used in conjunction with other order specifying commands. It causes TENEX to reverse the order of the directory listing.

d) Miscellaneous.

@@DELETED (FILES ONLY)%
TENEX prints data only for those files the user has deleted, but which have not yet been eliminated by EXPUNGE or a system backup operation. (See DELETE, UNDELETE, EXPUNGE this section.)

@@OUTPUT (TO FILE) file designator %
The output of DIRECTORY can be sent to other TENEX files or output devices, as well as the user's terminal. The file designator can specify a new or existing file; its name and extension, if omitted, default to "DIR" and ".DIR" respectively. Directory listings produced in this way will contain the current date and time on the same line as the directory name.

@@LPT%
Short for "OUTPUT (TO FILE) LPT:". Sends the output of DIRECTORY to the line printer.

e) Begin Listing. The following are equivalent; they terminate subcommand input and begin the listing.

```
@@BEGIN%
@@%
```

f) Examples. The following demonstrate some of the options available with DIRECTORY.

```
@DIRECTORY,
@@VERBOSE%
@@%

                                        PGS WRITE     READ      AUTHOR

 ALPHA.ABC;1;P777752;A123456            6   19-AUG-72 21-AUG-72 JONES
 BETA.  ;1;P7752ØØ;A56789               2   14-JUL-72  NOT READ SMITH
    .XYZ;2,1;P7752Ø2;A56789             11  1Ø-NOV-72 12-NOV-72 DOE

@

@DIRECTORY,%
@@LENGTH (IN BYTES)%
@@TIMES (AND DATES OF) CREATION%
@@BEGIN%

                  BYTES(SZ)     CREATION

 ALPHA.ABC;1      13693(7)      2-JUL-72  2:Ø1:47
 BETA.  ;1        36Ø2(7)       1Ø-JUL-72 1Ø:4Ø:56
```

```
     .XYZ;2      26Ø5Ø(7)      1Ø-NOV-72 1Ø:15:33
        ;1       26Ø41(7)      1Ø-NOV-72 1Ø:Ø5:21

@
```

In the first example a single entry sufficed for versions 1 and 2 of BETA.XYZ, since all information printed was identical for the two cases. The second example required two entries because the length (in bytes) and the time of creation differed slightly.

Errors: file designator errors
ILLEGAL DEVICE (doesn't have a directory)

2. QFD

To obtain a quick printout of all directory information concerning one or more disk files:

@QFD multiple disk file designators %

The information printed and format are equivalent to DIRECTORY plus the subcommands EVERYTHING, CRAM, and NO (HEADING).

```
@QFD ALPHA.ABC;1%
  <JONES>
  ALPHA.ABC;1;P777752;A123456 6 13693(7) 19-AUG-72
**15:46:3Ø 21-AUG-72 12:41:15 2-JUL-72 2:Ø1:47 JONES
@
```

TENEX printed the protection status, account, size, length, dates and times of creation, write, and read, and author, in that order.

3. COPY

To move information within TENEX:

@COPY (FILE LIST) multiple file designators (TO) file designator
[MESSAGE]%

The source file(s) must exist. TENEX takes the highest existing version of each file if none is specified. The destination designator can name an existing file, a new file, or a new version of an existing file. If the user

selects an existing file but omits the version number, TENEX supplies a number one greater than the highest existing version (IV-A-5). COPY accepts subcommands (described below) when the destination file is terminated with comma.

After termination of the second designator (space, altmode, comma, or carriage return) TENEX types one of the following messages:

```
[NEW FILE]
[NEW VERSION]
[OLD VERSION]
[OK] (for devices without file names)
```

depending on the file selected by the second designator, and then awaits a final confirming carriage return.

The effect of COPY depends entirely on the arguments given.

@COPY DTA1:ALPHA.ABC (TO) DATA.DAT;1 [NEW FILE]%

Information was transferred from DECtape 1 into a new disk file.

@COPY DATA.DAT;1 (TO) LPT: [OK]%

The file was listed on the line printer.

@COPY DTA1:BETA.XYZ (TO) D↑FATA.D↑FAT;1 [OLD VERSION]%

Another file was brought in from DECtape, replacing the contents of DATA.DAT;1. ↑F was used for recognition of DATA and .DAT so that the version could be entered by hand.

@COPY <SMITH>ALPHA.ABC;1 (TO) $ALPHA.ABC;2 [NEW VERSION]%

A file in directory <SMITH> was duplicated in the user's (Jones) connected directory. Beginning the destination file with altmode will default its name and extension to those of the last source file. TENEX picks a version one greater than the highest in the destination directory.

@COP PTR: PTP: [OK]%

A paper tape was duplicated.

Multiple Source Files

When multiple source files are designated, the information they contain is concatenated, in the order given, into the output file. If a list of files is entered, comma must terminate each file but the last. When * is used and

version omitted, TENEX will pick the highest version of each file selected.

```
@COPY B↑FETA.* (TO) ALPHA.ABC;3 [NEW VERSION]%
 BETA.;1
 BETA.XYZ;2
@
```

The contents of BETA.;1 and BETA.XYZ;2 were concatenated to form a new file, ALPHA.ABC;3.

```
@COPY A↑FLPHA.↑FABC;1,D$ATA.DAT;1 (TO) LPT: [OK]%
 ALPHA.ABC;1
 DATA.DAT;1
@
```

TENEX produced a combined listing of the two files.

Byte Size and COPY Subcommands

TENEX handles information in units called bytes whose size depends on the nature of the information and the information handling device. Generally the user need not concern himself with the question of byte size, as COPY, and the other TENEX file commands will nearly always make correct assumptions. However, the following should be noted.

Disk, magnetic tape, and DECtape files are physically broken up into 36 bit words, corresponding to the 36 bit word size of the PDP-1Ø computer, regardless of the kind of information they contain. 36 bit words are always transferred when copying between files stored on these devices.

Depending on its type, the information stored on these devices may occupy bytes equal to or less than the 36 bit word length. For example, binary information occupies 36 bit bytes, while ASCII characters occupy 7 bit bytes. When the byte size is less than 36 bits, bytes are packed left justified into 36 bit words with any unused bits on the right left vacant. For example, with ASCII information, five characters are packed into each word with the right most bit vacant. With disk files a record is kept of byte size; with magnetic or DECtape files, there is no such record.

Terminals and the line printer deal with 7 bit ASCII characters. 7 bit bytes are always transferred when copying between these devices. When the transfer is between one of these devices and disk, magnetic tape, or DECtape, TENEX will pack or unpack 7 bit bytes into or out of 36 bit words, as appropriate.

Because these devices are physically limited to one kind of information and one byte size, TENEX can always make correct assumptions about them when they appear in a file command. In general, the same is true of any other device restricted to a single byte size.

Certain devices can handle several types of information, each implying a different byte size and a different treatment. For example, the physical eight hole unit of paper tape information can be interpreted as containing an 8 bit byte, a 7 bit ASCII byte, or one sixth of a 36 bit binary byte.

With transfers to or from this kind of device, TENEX can sometimes make correct assumptions about the information involved. However, in other cases the situation is ambiguous, and TENEX must be told what to do.

To this end, COPY includes optional subcommands that specify the kind of information to be transferred. With 4/13/73 TENEX, these subcommands serve two main purposes. 1) They specify the information to be involved in a paper tape transfer, and 2) when COPYing from magnetic or DECtape to disk they can be used to establish the desired byte size record in the disk file. However, the intent of these subcommands is general, and they will apply to other multi-format devices that may be added to the system.

To modify COPY with subcommands, terminate the destination file designator with comma. After the confirming carriage return, TENEX will type "@@" and await subcommand input. A null subcommand (just carriage return) will initiate the COPY operation. The subcommands are:

@@ASCII%
Specifies 7 bit ASCII bytes. In particular, 1) declares a source or destination paper tape to be in ASCII format, and 2) stores a 7 bit byte size record with a destination disk file.

@@BINARY%
Specifies 36 bit binary bytes. Declares a paper tape to be in binary format; stores a 36 bit byte size record with a destination disk file.

@@IMAGE%
Specifies 8 bit image bytes. Causes an exact image transfer to or from paper tape in 8 bit bytes, each representing one eight hole line of paper tape. Allows punching and reading paper tapes in non-standard formats. With disk, magnetic tape or DECtape files, 8 bit bytes are packed four to a word with bits 31 to 35 unused. Stores an 8 bit byte size record with a

destination disk file.

@@IMAGE BINARY%
Same as IMAGE, except with paper tape this command suppresses the checksum processing normally done by TENEX. Not of interest to most users.

Examples:

```
@COPY PTR: (TO) DATA.DAT;2 [NEW VERSION],%
@@BINARY%
@@%

@COPY PTR: (TO)↑B FETA.MAC;1, [NEW FILE]%
@@ASCII%
@@%
```

COPY will always try to do something no matter what combination of files, devices, and subcommands is specified. However, the following conditions may arise. Certain subcommands will not work with certain file designators. For example, one cannot declare binary information in a COPY to the line printer. When this happens, TENEX ignores the subcommand and types the message:

```
[ILLEGAL MODE SUBCOMMAND BEING IGNORED]
```

In the absence of a subcommand, TENEX makes an assumption about byte size based on the devices involved, the byte size record on a disk file, and/or the extension of a named file. Usually, the assumption is unambiguous; however, with paper tape it may not be, and in this event TENEX types:

```
[YOU DIDN'T SPECIFY A PAPER TAPE FORMAT
WITH A SUBCOMMAND
SO ASCII/BINARY/IMAGE IS BEING ASSUMED]
```

Disk File Complications

Certain disk files (notably Save Core files - Section VII) may contain information taken from disjoint pages of core memory. As long as these files remain on the disk, their structure, i.e., the correspondence between file and memory pages is retained. However, when copied to any sequential (basically, non-disk) device, or when concatenated through the use of multiple file designators, the information in these files is compacted, and the original structure, or "holes", are lost because they get filled with zeros. Once this has happened, the file will no longer be "RUN"-able, even if it is copied back to a paging medium.

Files created with the SSAVE command (VII-A-2) have the additional complication of an "end of file pointer" beyond which additional pages may exist!

Should TENEX encounter an SSAVE file in a COPY to a sequential medium, the following message will be typed:

[HOLES IN FILE]

Use of ↑Z

When copying from the user's controlling terminal (TTY:) to some other file, all text following termination of the COPY command is taken as input data to be transferred. The (CTRL)-Z (↑Z) character serves as a terminator for the input text; typing ↑Z completes execution of COPY and returns control to TENEX.

Errors:
1) file designator errors
2) device CAN'T DO INPUT
3) device CAN'T DO OUTPUT
4) device CAN'T DO NORMAL MODE INPUT
5) device CAN'T DO NORMAL MODE OUTPUT

Errors 2 and 3 occur on misuse of an input only or output only device, for example an attempted COPY to PTR: or from LPT:. Errors 4 and 5 occur with special devices that can't be accessed through the file system in the normal manner. In 4/13/73 TENEX, magnetic tape units are in this category.

4. APPEND

This command adds the information in one or more source files to the end of a destination file. The source files are unaffected.

@APPEND multiple file designators (TO) existing
disk file designator %

Examples:

@APPEND D↑FATA.D↑FAT;1 (TO) D↑FATA.D↑FAT;2
[OLD VERSION]%

would add the contents of DATA.DAT;1 onto the end of DATA.DAT;2, leaving DATA.DAT;1 unchanged.

@APPEND B↑FETA.* TO A↑FLPHA.↑FABC.2 [OLD VERSION]%
BETA.;1
BETA.MAC;1
BETA.XYZ;2

```
@
```

The contents of the BETA files were concatenated and appended onto ALPHA.ABC;2.

```
@APPEND PTR: TO DATA.DAT;2 [OLD VERSION],%
@@BINARY%
@@%
```

Information from a paper tape was appended to DATA.DAT;2. The paper tape was declared to be in binary format. TENEX selected the highest existing version of the destination file.

The format and rules for COPY generally apply to APPEND, including the use of multiple file designators, byte size conventions, subcommands, the use of ↑Z, and warning messages.

Save Core Files can never be appended, even when a single source and destination are both on disk. Appending such files will always destroy their structure and lose their extra pages.

Errors: all COPY errors. Also:
DESTINATION FILE NOT ON DISK
FILE name CANNOT BE APPENDED TO
(file access protection violation)

5. RENAME

To change a file name:

```
@RENAME (EXISTING FILE) existing file designator
     (TO BE) file designator %
```

Both file designators must refer to the same device, and the device must carry named files. If the second file designator specifies an existing file, the file's contents are expunged, and its name given over to the first file. After the second file designator TENEX types one of the messages (such as [NEW FILE]), listed under COPY, and then awaits final confirmation. Example:

```
@RENAME (EXISTING FILE) DATA.DAT;2 (TO BE) SAVE.DAT;1
     [NEW FILE]%
```

Errors: file designator errors
FILES NOT ON SAME DEVICE

6. ACCOUNT

To change the account that will receive storage changes for a disk file:

```
@ACCOUNT (OF FILE) existing disk file designator
     (IS) account %
```

The Account file descriptor is covered in IV-C-3. Depending on his status (III-A-3) the user must enter a five or six digit number or an alphanumeric account name. Example:

```
@ACCOUNT (OF FILE) SAVE.DAT;1 (IS) 12345%
```

Errors: file designator errors
DISK FILES ONLY
"?" if account must be numeric but isn't

7. PROTECTION

To change the protection descriptor for a file:

```
@PROTECTION (OF FILE) existing disk file designator
     (IS) protection %
```

"Protection" is a 6-digit octal number as described in IV-D-2. Example:

```
@PROTECTION (OF FILE) BETA.;1 (IS) 770000%
```

This would restrict file access to the user.

Errors: file designator errors
DISK FILES ONLY
"?" if protection not octal number
YOU CAN'T MAKE PROTECTION INVISIBLE (20000-BIT)

The same effect can be achieved with RENAME. For example, the following is equivalent to the example above:

```
@RENAME BETA.;1 (TO BE) BETA.;1;P770000%
```

8. DELETE

To eliminate one or more files:

```
@DELETE multiple file designators %
```

If on a device other than disk, the files are eliminated. If on disk, they are marked as deleted and become inaccessible to TENEX commands except UNDELETE, EXPUNGE (below) and DIRECTORY. After deletion, disk files remain on

the disk and can be UNDELETEd until EXPUNGEd or until the next routine disk backup operation. (See system operators for disk backup schedule.) If a file designator is entered without version number, TENEX will delete the lowest non-deleted version. Examples:

```
@DELETE SAVE.DAT;1%

@DELETE B↑FETA.*;*
 BETA.;1
 BETA.MAC;1
 BETA.XYZ;1
 BETA.XYZ;2
@
```

If a file designator list is specified, the individual designators can be separated with comma or space, and altmode can be used anywhere in the list. Since name and extension default to those of the previous entry, they can be omitted when several versions of the same file are to be deleted:

```
@DELETE DTA1:B$ETA.XYZ ,A$LPHA.ABC;1 ,;3%
 DTA1:BETA.XYZ
 ALPHA.ABC;1
 ALPHA.ABC;3
@
```

Errors: file designator errors
PROTECTION VIOLATION
CAN'T DELETE ARCHIVE-PENDING FILE

9. UNDELETE

To revive DELETEd disk files:

```
@UNDELETE multiple deleted file designators %
```

The file(s) are restored to normal status; they are once again accessible to executive commands. Multiple file designators can be specified with the format options of DELETE available for entering file designator lists. In the case of multiple files, any non-deleted files are ignored, and their designators are not printed. Examples:

```
@UNDELETE SAVE.DAT;1%

@UNDELETE A↑FLPHA.*;*%
 ALPHA.ABC;1
 ALPHA.ABC;3
@
```

Errors: file designator errors
NOT DELETED

10. EXPUNGE

To remove permanently any DELETED files in the connected directory:

@EXPUNGE%

This is an abbreviation for:

@EXPUNGE DELETED (FILES)%

Other possible keywords for modifying EXPUNGE are: SCRATCH, TEMPORARY, and ALL. The last of these specifies all deleted, scratch and temporary files. After an EXPUNGE operation, it will be impossible to UNDELETE any of these files.

Errors: none

11. CLEAR

To remove all files on a private disk pack or DECtape:

@CLEAR (DIRECTORY OF DEVICE) device name
[CONFIRM]%

Because of its possibly disastrous consequences, this command always requires a confirming carriage return on the next line. Example:

@CLEAR (DIRECTORY OF DEVICE) DTA3:
[CONFIRM]%

Errors: "?" if no such device (type 2)
DECtapes only
device: NOT AVAILABLE
device: ASSIGNED TO JOB n
device: NOT MOUNTED

12. SHUT

As discussed in IV-D-2, TENEX files must be opened prior to manipulation by user programs or other mechanisms within TENEX. To close all files opened by a subsystem or user program:

@SHUT (ALL OPEN FILES)%

Errors: none. No complaint if no files open.

13. JFNCLOSE

When opened, a TENEX file is assigned a Job File Number (JFN) (IV-D-2). To close a specific opening of a file:

@JFNCLOSE jfn %

A list of currently assigned JFN's may be obtained with the FILSTAT command (VIII-9). Example:

@JFNCLOSE 3%

Errors: "?" for no such JFN assigned.

Note: SHUT and JFNCLOSE generally need only be applied in unusual cases, such as program debugging. Files opened by TENEX commands such as COPY or APPEND are generally closed as part of command execution. Files opened by subsystems or user programs are generally closed by the subsystem or program in question. Any such files are also closed by the TENEX commands GET, RUN, START or RESET (VII). JFNCLOSE is more powerful than SHUT in that it will close files left open by (rare) TENEX command failures, as well as files opened by subsystems or user programs.

14. CONNECT

As explained in IV-A-2, when a user logs in to TENEX, the system initially sets up access to his primary disk directory. Subsequent to logging in, when the user omits directory name from an executive command, TENEX takes his primary directory name as default value. To shift access to a different disk directory, either one's own or someone else's:

@CONNECT (TO DIRECTORY) directory name
(PASSWORD) password %

Subsequent to this command, TENEX will take the new directory name as default value when it is omitted from executive commands. As well as directory name, the user must specify the password associated with the new directory. The password may be omitted if the user has "group access" to the directory. (As with all commands which require a password, the user should terminate the password input with a space if a half duplex scope terminal is in use.)

CONNECT triggers an allocation check for both the previously connected directory and the directory being connected to. If either or both are over the limit, a message will be typed. The TENEX Operations Manager may be contacted in order to secure an increase in disk space.

```
@CONNECT (TO DIRECTORY) SMITH (PASSWORD) %
@DIRECTORY
 ABC.FOO;1
@
```

It is always possible, of course, to access any file or directory to which one has been granted access privilege, by supplying its name directly in executive commands:

```
@CONNECT (TO DIRECTORY) JONES(PASSWORD)%
@DIRECTORY <SMITH>%
 ABC.FOO;1
@
```

Errors: "?" for no such directory (type 2)
INCORRECT PASSWORD

15. LIST

To print out the contents of symbolic files on the line printer at the computer center:

```
@LIST (FILE) multiple file designators %
```

Examples:

```
@LIST (FILE) FOO.MAC;1%

@LIST (FILE) ALPHA.ABC,BETA.XYZ;1%

@LIST (FILE) PTR:%
```

The second example shows how more than one file can be listed with one command; the third would list a paper tape.

Printing may not occur instantly after a "LIST" command. The files are usually "spooled" for future listing by a system job.

Each page of the listing is normally headed with the file name, a date and time, and the page number. The date and time when last written is used for disc files; the date and time of listing is used for other files.

A new page is started whenever a form feed character ((CTRL)-L) is encountered in the file. If the file contains more lines between two form feeds than will fit one page of printout, TENEX will divide that page, at what it considers reasonable places, into as many pages as necessary. The additional pages created by these divisions are numbered "n:1", "n:2", etc. For example, a file containing no form feeds at all will be printed with pages numbered 1, 1:1, 1:2, etc.

Any control characters (except NULL) encountered in the file, including form feeds (↑L's) are printed as "↑" followed by a letter.

If a comma is typed between the last file name and the final carriage return in a LIST command, subcommands will be accepted. As with other commands accepting subcommands, subcommand input is terminated and execution initiated by typing just a carriage return. The subcommands are:

@@PAGES n,n,n-n... %
Allows listing only selected pages or groups of pages from file.
Example: @@PAGES 1,5,21-27,32-33,58%

@@HEADING any text %
Allows specifying any heading you wish to be used at the top of each page of listing. The text may be terminated with carriage return or altmode; multiline headings may be given by using the continuation character "&". No heading at all may be specified by typing just HEADING (carriage return). If HEADING is given, the page numbers are still printed unless the following subcommand is also given.

@@NO (PAGE NUMBERS)%
Suppresses printing of page numbers. If heading is also suppressed, this makes three more lines available for printout.

@@SPACING n %
SPACING 2 causes listing to be double-spaced, SPACING 3 causes it to be triple-spaced, etc.

@@WIDTH n %
Specifies maximum number of columns to use. When a line longer than the width is encountered, its continuation on the next line is prefixed with "**". Default width is 80 except 72 when output is to a terminal. Thus when you wish to make a listing on wide line printer paper, you must give WIDTH 132.

@@LENGTH n %
Gives maximum number of lines per page. Default is 60.

@@INDICATE (NULLS BY ↑@)%
Shows NULLs (octal 0) in the file by the graphic "↑@".

@@OUTPUT (TO FILE) file designator %
Sends output to specified device, terminal, disc file, etc, instead of to the line printer.

In addition, the following commands are available. They are of use when the line printer spooler is not operational.

@@DETACH (BEFORE LISTING)
This subcommand causes the user's job to be detached as soon as the listing is started. He may then LOGIN again to continue working.

@@LOGOUT (AFTER LISTING)%
This subcommand may be used in addition to the DETACH option, and will cause the job to automatically log itself out when the listing has finished.

Errors: file designator errors (see IV-E)
LPT: NOT MOUNTED (off line)

16. TYPE

To print symbolic files on your terminal:

@TYPE (FILE) multiple file designators %

Identical to LIST (just described) except for the use of the terminal in place of the line printer.

If you wish printout to be properly positioned on terminal paper which has perforations between pages, initialize the paper position before TYPEing by typing a form feed (CTRL-L) then setting the paper manually to the top of a page. Then enter your TYPE command, using a form feed in place of the final carriage return.

At the end of each page, paper is fed to the top of the next page with a form feed character. TENEX normally simulates a form feed by typing the appropriate number of carriage returns, but users with suitably equipped sprocket-feed terminals may obtain listings faster by giving the FORMFEED command (see IX-2) prior to TYPE.

Some users with friction-feed terminals will find that their terminal does not feed paper accurately, so that the listing "drifts" upward or downward on the page. Such users will find this additional subcommand useful:

@@PAUSE (BETWEEN PAGES)%
Causes TENEX to ring the terminal bell and wait for you to type a carriage return at the top of each page, giving you a chance to readjust the paper position manually when necessary.

Errors: see LIST

17. ARCHIVE

To mark a file(s) for future archiving on (two separate) magnetic tapes:

@ARCHIVE multiple file designator%

The archiving will not be done instantaneously because it requires the TENEX Operations Staff to mount special tapes and run the "backup system" program. Typically this might be done twice a week.

Note: Not all TENEX installations support the ARCHIVING system. Check with the TENEX Operations Manager before using this command.

ARCHIVE has several subcommands:

@@DEFERRED%
Archive the file(s) at the next regular archival run of of the backup system. This is the default.

@@DELETE%
The file(s) are to be deleted after archiving. This is the default.

@@DON'T ARCHIVE%
Prevents the file(s) from ever being archived.

@@DON'T DELETE
Specifies that the files are not to be deleted after

archiving.

@@IMMEDIATE%
Causes the file(s) to be moved out of the specified directory to a special system directory and marked for archiving. This gives the illusion of instantaneous archiving.
NOT IMPLEMENTED YET

@@RESET%
Clears all archival attributes of the file. This will not "unarchive" a previously archived file.

@@STATUS%
Types the archival status of the file(s).

18. INTERROGATE

To determine which tapes a file is archived on:

@INTERROGATE filename%

Replies with either "THAT FILE NOT ARCHIVED" or "ARCHIVED ON BBN TAPES nnn AND mmm". In the second case, "DO YOU WANT IT RETRIEVED?" will be asked. If the answer is YES, the TENEX Operations Staff will automatically be notified and the file will be recovered (usually within a few hours). As files are retrieved, their owners are notified automatically via MAIL.

19. EPHEMERAL

"Ephemeral subsystems" ("ephemerons") are runnable files, ususally with .SAV as an extension, which the EXEC will execute in a special address space separate from the one which the user normally manipulates with commands like MEMSTAT, SSAVE, etc. Control-C while an ephemeron is running or an error during its execution causes it to disappear without a trace.

Ephemerons may be stored in any directory. They are invoked in the same way as any program -- by simply mentioning the name of one. If the name is terminated by a space, no confirming carriage return need by typed and the ephemeron is started immediately. Thus, the ephemeral program may parse the rest of the command line in any way it chooses. Typically, this will be using the various JSYS's such as GTJFN for file names or NIN for numbers. Because of this, and the way ↑C is handled, ephemerons appear as extensions

of the EXEC command language.

To declare a file as possessing the ephemeral property:

@EPHEMERAL file name%

Example:

@EPH <SUBSYS>DELVER.SAV%

Declaring a file as ephemeral requires that the person doing the command possess ownership rights on the file.

2Ø. NOT EPHEMERAL

To convert an ephemeron into an ordinary .SAV file:

@NOT EPHEMERAL file name%

After this command, the execution of the file will take place in the normal user address space beneath the EXEC. Requires ownership rights.

G. FUTURE ADDITIONS

Subsequent versions of TENEX will have several new File Commands. One command, DEFINE, will allow users to create synonyms for a file name, without having to RENAME the original file.

V. DEVICE HANDLING

A. DEVICE COMMANDS

Most input/output devices allow access by only one user at a time. To provide orderly access, the commands in this section allow the user to reserve a device prior to accessing it, and to release a reserved device when it is no longer needed. These commands apply in particular to magnetic tape, disk packs, DECtape units, and the paper tape punch and reader, which must generally be reserved before and released after use through the ASSIGN and DEASSIGN commands below.

With other devices the situation varies. Controlling terminal lines are assigned in orderly fashion to entering users by the system operators or the dial up equipment. However, a user job can access additional terminals beyond the controlling terminal for input/output purposes, and when used this way, terminal lines should be ASSIGNed and DEASSIGNed. Although the line printer can take only one user at a time, it need not be reserved in advance, since TENEX handles the queuing of line printer output requests internally.

As well as the need for reservations, disk packs, DECtapes and magnetic tapes carry file directories that must be communicated to TENEX before their files can be read or written. When a tape unit is ASSIGNed, TENEX reads the directory from the tape or pack (if any) it carries as part of the assignment process.

Two additional commands allow further operations on tapes and tape directories. MOUNT will read the directory from a tape unit and enable access to its files. This is useful when no tape was loaded at the time the unit was ASSIGNed or when a subsequent tape is to be accessed on the same unit. Since MOUNTing an unassigned unit does not block access from other jobs, MOUNT can also be used instead of ASSIGN when two or more jobs need simultaneous access to the same tape unit. UNMOUNT will block computer access to a tape unit while a new tape is being loaded. This prevents inadvertent access to the unit until the new directory has been properly transmitted to TENEX.

1. ASSIGN

To reserve a device and read its directory (if any):

@ASSIGN device name%

Examples:

@ASSIGN PTR:%
@ASSIGN DTA3:%

Errors: "?" if no such device (type 2)
device: CANNOT BE ASSIGNED
device: NOT AVAILABLE (not assigned,
but in use by another job)
device: ALREADY ASSIGNED TO JOB n
YOU CAN'T ASSIGN YOUR CONTROLLING TERMINAL
SYNONYMS NOT IMPLEMENTED YET

2. DEASSIGN

To release a previously ASSIGNed I/O device:

@DEASSIGN device name%

Examples:

@DEASSIGN PTR:%
@DEASSIGN DTA3:%

Errors: "?" if no such device (type 2)
device: NOT ASSIGNED
device: NOT ASSIGNED TO YOU

3. MOUNT

To cause TENEX to read the directory from a magnetic or DECtape unit, and enable access to its files:

@MOUNT device name %

Examples:

@MOUNT DTA3:%

Errors: "?" if no such device (type 2)
device: NOT A MOUNTABLE DEVICE
device: NOT AVAILABLE
device: ASSIGNED TO JOB n

4. UNMOUNT

To block access to a tape unit:

@UNMOUNT device name %

Example:

@UNMOUNT DTA3:%

Errors: "?" if no such device (type 2)
device: NOT A MOUNTABLE DEVICE
device: NOT AVAILABLE
device: ASSIGNED TO JOB n
device: NOT MOUNTED

5. REWIND

To rewind either a DECtape or magnetic tape to the logical beginning of tape:

@REWIND device name %

Example:

@REWIND MTA3:%

Errors: DEVICE ASSIGNED TO JOB n
DEVICE NOT AVAILABLE

6. UNLOAD

To unload ("flap") a DECtape or magnetic tape. This is especially useful when operating from a remote location when an operator is not available to remove a tape which is write-enabled or which has confidential information stored on it.

@UNLOAD device name %

Example:

@UNLOAD DTA3:%

ERRORS: DEVICE ASSIGNED TO JOB n
DEVICE NOT AVAILABLE

B. FUTURE ADDITIONS

Future TENEX will permit one to refer to I/O devices by "logical names" in user programs, equating an appropriate physical device to each such name prior to running the program. For example, the logical name "SOURCE" might be equated with DTA1: on one occasion, DTA3: on another, and PTR: on a third. This "synonym feature" will permit a program to access different devices on different occasions without needing to be rewritten. It will avoid the problems that otherwise occur when the expected I/O device is assigned to another job when the program is to be run. When the synonym feature is implemented, ASSIGN will take on the form:

@ASSIGN device name (AS) [logical name] %

where logical name is an optional argument. An attempt to use this form at present will trigger the error message: "NOT IMPLEMENTED YET".

VI. SUBSYSTEM CONTROL

TENEX contains a number of subsystems, each stored in the system as a disk file, and designated in the usual way by file name,extension;version number. The collection of subsystem file designators is treated as a special group of TENEX commands. To activate a subsystem, type its file designator instead of an ordinary TENEX command, at any point where TENEX is awaiting command input. Terminate with carriage return. TENEX will perform recognition on subsystem designators as explained in IV-B, selecting the highest version number if more than one version exists.

Example:

@TECO.SA;47%

would activate the TECO subsystem. Control will now pass from the executive to the subsystem, and further communication will take place in the language of the particular subsystem activated. Also, the way to deactivate the subsystem and return control to the executive will depend on the conventions of the particular subsystem. However, control can always be returned to the executive by typing one or more control-C's (↑C).

The table below lists the major subsystems and their functions. In addition there are a number of other subsystems serving less common purposes. All subsystems are filed together in one directory with the name <SUBSYS>. At any time one can obtain a complete list of all subsystems by:

@DIRECTORY <SUBSYS>%

Note, however, that the directory name <SUBSYS> need not be given when activating subsystems as above.

Full documentation on the properties and operation of the subsystems is contained in the TENEX USER'S GUIDE which is available from BBN. In the table that follows, version numbers have been omitted, since these change from time to time as the subsystems are updated.

NAME	DESCRIPTION
1. Conversational computation languages	
BASIC.SAV	Basic algebraic language compiler
2. Symbol manipulation languages	
LISP.SAV	BBN LISP language
SNOBOL.SAV	SNOBOL4 language
3. Text editing languages	
TECO.SAV	Character-oriented text editor
RUNOFF.SAV	Output formatted text
SORT.SAV	Column-oriented, COBOL-based sort program
4. Assemblers and compilers	
F4Ø.SAV	Fortran IV compiler
MACRO.SAV	Standard DEC assembler
FAIL.SAV	A single-pass assembler
BLISS.SAV	Carnegie system implementation language compiler
SAIL.SAV	Stanford extended ALGOL compiler
MIDAS.SAV	M.I.T. assembler
PPL.SAV	A Harvard interactive, extensible programming language
PALX.SAV	Assembler for the PDP-8
ALGOL.SAV	DEC ALGOL compiler
COBOL.SAV	DEC COBOL compiler
BCPL.SAV	Basic Compiler Programming Language
5. Loaders	
LOADER.SAV	Linking loader for output of all PDP-1Ø assemblers and compilers
6. Utility programs	
FUDGE2.SAV	Build and update library program files
BSYS.SAV	Backup system
CCL.SAV	Concise Command Language
RUNFIL.SAV	Program to take command stream from file instead of TTY:
DO.SAV	A version of RUNFIL which allows dummy arguments

COPYM.SAV	Copies multiple groups of files
MTACPY.SAV	Copies magnetic tapes
DTACOPY.SAV	Copies DECtapes
MINCOP.SAV	Copies DUMPER format tapes
DUMPER.SAV	A magnetic tape backup system
DSKAGE.SAV	Generates age profile histogram of disk files
DELVER.SAV	Management of file versions

7. Debugging programs

SDDT.SAV	A "visible" DDT with built-in TENEX symbols
UDDT.SAV	A DDT which picks up the user's symbols Invoked by the DDT command, VII-A-14.
IDDT.SAV	An "invisible DDT" for debugging programs in an inferior fork Invoked by the IDDT command.

8. Cross-reference and comparison programs

CREF.SAV	Cross-reference generator: assembly program symbols
SRCCOM.SAV	Compare and list symbolic files
BINCOM.SAV	Compare binary files
TYPREL.SAV	Analyze contents of relocatable binary files
GLOB.SAV	Produces global cross-reference from .REL files.
FILCOM.SAV	A DEC program similar to a combination of SRCCOM and BINCOM
TYPBIN.SAV	Types .REL file block types
FRKCOM.SAV	Compares an address space with a .SAV file.

9. Conversion programs

CONVERT.SAV	Convert SDS-94Ø formatted tapes to and from TENEX files
FIOCNV.SAV	Convert PDP-1Ø symbolic files to paper tapes in Friden or Dura code
RELRIM.SAV	Convert relocatable binary files to RIM mode paper tapes
FLIST.SAV	Does column 1 format conversion for FORTRAN output
BCDTAP.SAV	handles BCD magnetic tapes

IMGPTP	Produces a paper tape from a .SAV file
TAPCVT.SAV	Converts IBM card-image tapes to TENEX format
LBLOCK.SAV	Line-blocks text files for FORTRAN compatibility

1Ø. Applications programs

ECAP.SAV	Electronic cicuit analysis program
FLOW.SAV	Flow charts FORTRAN source programs
LPTPLT.SAV	Uses the line printer for plotting applications
TTYTST.SAV	Terminal test program
TTYTRB.SAV	Used to report terminal, line, and scanner troubles.
GRIPE.SAV	Used to make causal comments regarding TENEX programs.
SNDMSG.SAV	Used to send mail to other users.
READMAIL.SAV	Used to receive mail.

11. Programs which use the ARPANET

TELNET.SAV	Makes terminal connection to remote sites
FTP.SAV	Transfers files to and from remote sites
NETSTAT.SAV	Types status of the ARPANET
MAILER.SAV	Delivers mail to remote sites
RJS.SAV	Remote job submission to the the UCLA 36Ø/91
RSEXEC.SAV	Network-wide Resource-Sharing Exec

12. Miscellaneous

WATCH.SAV	System statistics
<HACKS>CHESS.SAV	Chess playing program
<HACKS>DOCTOR.SAV	Simulated psychiatrist
<HACKS>JOTTO.SAV	Plays a word game
<HACKS>LIFE.SAV	Displays successive colonies of "LIFE"
<HACKS>SNOW.SAV	Makes banners on the line printer

VII. PROGRAM CONTROL AND DEBUGGING

TENEX contains a special class of files and a number of commands that assist in saving, manipulating, and debugging programs. TENEX Save Core (extension.SAV) files contain information taken from core memory. Their main intent is to preserve completed binary programs. A Save Core file contains an image of those portions of memory selected by the user through the SAVE or SSAVE commands described below.

The program in a Save Core file will usually contain a main, initializing start up location. It may also contain a re-entry location that serves as a convenient place for restarting the program after an interruption (for example by ↑C). By convention, these locations are generally placed in contiguous addresses to form what is known as the program's "entry vector".

The address and length of the entry vector can be declared as part of the loading process when programs are set up through the loader subsystem, or by the ENTRY VECTOR command described in this section. When a Save Core file is created, the location and length of the entry vector is stored in the file along with the program. When the file is reloaded into core this information is communicated to TENEX for use by the RUN, START, and REENTER commands described below.

Save Core files are created by SAVE and SSAVE. The data they contain can be loaded back into the memory locations it came from through GET, MERGE, or RUN. GET and MERGE will load the file without starting it; RUN will start execution at the starting address stored with the file. In addition, a Save Core file, once placed back in core, can be started by START (at the starting location), REENTER (at the reentry point) or GOTO (at an arbitrary location specified by the user). These commands, together with a number of other TENEX program control and and debugging commands, are described below.

Virtually all commands in this section operate on an address space. If the user has more than one, the FORK command (VII-A-17) may be used to direct the EXEC's attention to a specific one.

A. PROGRAM CONTROL AND DEBUGGING COMMANDS

1. SAVE

To create a Save Core file:

@SAVE (CORE FROM) n (TO) n, (FROM) n (TO) n, . . .
(FROM) n (TO) n (ON) file designator [message]%

In the above, "n" stands for an octal core address (from 2Ø through 777777). Each pair of addresses denotes a range of core locations to be saved. The first member of each pair is terminated with altmode or space. With altmode, TENEX will echo the "(TO)", otherwise not. The second address of a pair can be terminated by space, altmode or comma. Comma indicates another pair to follow. After the comma the user can enter another address immediately, or type altmode, in which case TENEX will type (FROM) before awaiting the next address. Space or altmode instead of comma (causing nothing or "(ON)" to be echoed, respectively) denote the end of the address list and indicate that a file designator is to follow. Any legitimate file designator is legal. If the user terminates file name with altmode, TENEX will default the extension to SAV;. Following the file designator, TENEX will type one of the standard file messages listed under COPY (IV-F-3). For example:

@SAVE (CORE FROM) 123 (TO) 777,$ (FROM) 1ØØØØ (TO) &
17777 (ON) DTA1:PROG1.SAV [NEW FILE]%

would create a new DECtape file, "PROG1.SAV" containing the contents of core locations 123(8) through 777(8) and 1ØØØØ(8) through 17777(8), plus the location and length of the entry vector, if any. Note the use of the & character to continue the command on the next line. (See II-B-8)

Had the user substituted space for the altmode terminator used by convention in this manual, and omitted altmode after the comma, the command would look like:

@SAV 123 777, 1ØØØØ 17777 DTA1:PROG1.SAV [NEW FILE]%

The user can save all, or the bottom or top portion of core by defaulting either or both members of the first address pair with altmode. In this case, TENEX will select (and type back) 2Ø for the first address, 777777 for the second. Specifying addresses for which no core is assigned does not produce an error; any assigned core is saved and the rest of the request is ignored.

SAVE does not save the PDP-1Ø accumulators (locations Ø to 17), nor does it save blocks of zero memory.

For example,

```
@SAVE (CORE FROM)$2Ø (TO) $777777 (ON) &
DTA1:LONGDUMP.SAV;1 [NEW FILE]%
```

would save all currently assigned core while

```
@SAVE (CORE FROM) $2Ø (TO) 1ØØØØ (ON) PTP:%
```

would save any core below 1ØØØØ on paper tape.

Other examples of save core commands follow:

```
@SAVE (CORE FROM) 1ØØØ (TO) 7777 (ON) &
ALPHA.SAV;1 [NEW FILE]%

@SAVE (CORE FROM) 2Ø (TO) 21 (ON) &
B↑FETA.XYZ;3 [NEW VERSION]%
```

Errors: NO PROGRAM
"?" for upper limit less than lower
"?" for number greater than 777777
file designator errors (see IV-E)

2. SSAVE

TENEX contains a mechanism that allows pages of core memory to be shared by concurrently running jobs. This mechanism is aimed chiefly at subsystems that may be in simultaneous use by more than one TENEX user. Shared pages allow efficient use of core memory, reduce system overhead due to page swapping, and speed up the process of loading files into core. For a complete discussion of page sharing in TENEX, the user is directed to various technical papers available from Bolt Beranek and Newman Inc.

The SSAVE command allows the user to create files containing sharable memory pages. When loaded into core, each page will be shared by any jobs attempting to access the file in question.

In order for sharing to work, the shared pages must contain "pure code", that is, no storage locations within them should be changed during execution. In this case sharing will proceed in normal fashion. Should a page fail to contain pure code, then a private copy will be made for each job that changes its contents. This exploits the sharing mechanism as much as possible, but allows for the case where sharing conventions are not adhered to.

SSAVE differs from SAVE in that it creates sharable memory pages; it also differs in that the memory to be saved is specified in octal page numbers from Ø to 777, instead of core addresses. The format of SSAVE is:

@SSAVE (PAGES FROM) n (TO) n, (FROM) n (TO) n,...
(FROM) n (TO) n (ON) file designator [Message]&

In general, the format rules for SSAVE are identical to those of SAVE. The first pair of pages can be defaulted (by altmode) to Ø and 777 respectively.

For example,

@SSAVE (PAGES FROM) 3 (TO) 7, (FROM) 4ØØ (TO) 477 (ON) &
SHRFIL.SAV;1 [NEW FILE]&

Sharable Save Core Files can exist only on disk. Specifying a device other than disk with SSAVE or attempting to COPY a file created with SSAVE to another device will yield an unusable file. Non-existent pages of the address space being SSAVE'd do not consume space in the file. However, a page full of zeros will be saved in the file.

Note that SSAVE will not save the PDP-1Ø accumulators (locations Ø through 17) even if page Ø of the address space is SSAVE'd.

Errors: See SAVE except:
"?" for number greater than 777

3. ENTRY VECTOR

To establish an entry vector for a program currently in core, or to change the present entry vector:

@ENTRY (VECTOR LOCATION) octal address (LENGTH)
[octal number] &

The address specifies where the entry vector is located in core; the length specifies its length in memory words. Lengths of Ø, 1, and 2 are normal.*

Length Ø specifies that no entry vector exists. Length 1 specifies a one word entry vector which is the program's start location; length 2 specifies a two word vector consisting of the program's start and re-entry locations in that order. If the length field is defaulted, TENEX will

---

*To specify entry vectors compatible with the DEC 1Ø/5Ø time sharing system, length should be set to 254ØØØ.

assume and type a length of 1.

Examples:

@ENTRY (VECTOR LOCATION) 1776 (LENGTH) 2%

A two word entry vector was declared at location 1776.

@ENTRY (VECTOR LOCATION) 400 (LENGTH) $1%

A one word entry vector at 400.

Errors: NO PROGRAM
"?" for length greater than 777

4. GET

To load a Save Core file back into core memory:

@GET file designator %

GET first clears the user's job by doing a RESET (see below). Next, it reloads the Save Core information back into the addresses it came from. The location and length of the entry vector, if any, is also read from the file and placed in system storage.

GET-ing an SSAVE or SAVE file will not change the fork's accumulators.

Any legal file designator can be specified with GET, but if the file got is not in fact a Save Core file, unpredictable results will occur. GET can be used with files created with either SAVE or SSAVE. With GET, TENEX will default to a .SAV extension, if one exists. Otherwise the recognition and default rules of IV-B-3 apply. Version number will default to the highest existing number.

Examples:

@GET PTR:%
@GET ALPHA.SAV;1%
@GET BETA.;1%

In the last example, no .SAV file existed, but TENEX found a null extension.

Errors: file designator errors
BAD CORE SAVE FILE FORMAT
SYSTEM SPECIAL PAGES TABLE FULL

5. MERGE

To load a Save Core file back into memory, merging it with any information already there:

@MERGE file designator %

MERGE is similar to GET except 1) that TENEX does not RESET before loading the indicated file, and 2) if an entry vector was already present, the new one is not loaded. This means that currently assigned memory is generally left intact prior to the MERGE operation. If the contents of an SSAVE file occupy memory pages disjoint from those already assigned, then new core is assigned and the result is a simple merger of the file with what is already in core. If the file information occupies pages overlapping those currently assigned, then the overlapped pages will be cleared, and then replaced with pages from the file.

On the other hand, MERGE-ing a SAVE file will never clear any locations since zeros are not saved in SAVE files.

If no memory was assigned at the time of the MERGE, then, as with GET, new memory is simply assigned as needed.

Provided an entry vector was present prior to the MERGE, its addresses will continue to govern if the user does a START or REENTER (see below).

As with GET, any file designator is legal, but only Save Core files are meaningful. Also, with MERGE, TENEX handles file extension recognition in the same way as with GET.

Examples:

```
@GET ALPHA.SAV;1%
@MERGE BETA.XYZ;3%
```

Referring back to the SAVE examples, the got file would occupy 1000(8) to 7777(8) and the merged file 20 and 21(8). The start and reenter addresses from ALPHA would continue to govern after the MERGE.

```
@GET BETA.XYZ;3%
@MERGE ALPHA.SAV;1%
```

This would put the same information in memory, but now the entry vector from BETA would govern.

```
@GET DTA1:LONGDUMP.SAV;1%
@MERGE ALPHA.SAV;1%
```

Here the contents of ALPHA would overwrite the 1ØØØ(8) through 7777(8) portion of LONGDUMP.

```
@GET BETA.XYZ;3%
@MERGE DTA1:LONGDUMP.SAV;1%
```

Here, LONGDUMP (2Ø - 777777(8)) might wipe out the contents of BETA, but the start and reenter addresses from BETA will continue to apply.

Errors: see GET

6. RUN

To load and start a Save Core file:

```
@RUN file designator %
```

Run does everything GET does, and behaves in exactly the same way with respect to file designators and file extensions. Having got the indicated file and set it up in core, RUN then starts it at the starting address.

Examples:

```
@RUN PTR:%

@RUN ALPHA.SAV;1%
```

Errors: see GET, also:
NO START ADDRESS

The following three commands, START, REENTER, and GOTO, all assume that a program has been placed in core, either through the LOADER subsystem or through GET, MERGE, or RUN. The program may have been freshly loaded, have run to completion, or had its running interrupted by ↑C or a monitor trap. Of these three commands, START and REENTER cause execution to begin at the starting and reentry addresses, respectively, while GOTO allows the user to select an arbitrary starting location.

A fourth command, CONTINUE, provides a means for restarting a program after an interruption by ↑C. In this case, TENEX remembers where execution left off at the time of the ↑C, and execution begins again at this point. This is true even though the EXEC may have been pointed at a different fork using the FORK command (see VII-A-17).

RUN is assumed as the default command. Thus if the name of a save file is mentioned, the EXEC will look for the program in directory <SUBSYS>, then in the connected directory, and finally in the user's login directory. Refer to II-A-6-b for more details.

Ephemerons are never RUN in the user's address space. They are almost indistinguishable from commands to the EXEC itself. Thus, START, REENTER, GOTO, CONTINUE, Examine(/), and Deposit(\) have no meaning with regard to "ephemeral subystems". Refer to section IV-F-7.

7. START

To begin program execution at the starting location in the entry vector:

@START%

Prior to beginning execution, START closes all currently open files.

Errors: NO PROGRAM
NO START ADDRESS

8. REENTER

To begin program execution at the reentry address:

@REENTER%

The idea behind reenter is to provide a graceful means for restarting programs without having to repeat a possibly lengthy initialization sequence. This initialization may have taken place in an earlier session or before an interruption during the current session.

Errors: NO PROGRAM
NO REENTER ADDRESS

9. GOTO

To begin program execution at an arbitrary location:

@GOTO octal address%

The address must lie between Ø and 777777(8). Core memory must be assigned at this location and the address should, obviously, be a meaningful starting location for the program.

Examples:

```
@GOTO 123%
@GOTO 1ØØØ%
```

Errors: NO PROGRAM
NO SUCH PAGE
CAN'T EXECUTE THAT PAGE

1Ø. CONTINUE

To continue execution of a program interrupted by ↑C:

```
@CONTINUE%
```

When the user interrupts a program by ↑C, TENEX saves all volatile information, such as the contents of the program counter, accumulators, and other active registers. Commands which do not destroy the address space (i.e., other than GET, RESET, etc.) may be executed. Then, CONTINUE causes the saved information to be restored and execution to be resumed. The net effect is as though the interruption had never taken place.

Errors: NO PROGRAM
PROGRAM HASN'T BEEN RUN
NOT INTERRUPTED

11. RESET

To clear the job to which the user's terminal is currently attached:

```
@RESET%
```

RESET 1) releases any currently assigned core memory, 2) releases any subsystem currently in use, 3) closes any files that may be open. Other than assigned input/output devices (Section V) (which can only be released through DEASSIGN or LOGOUT), the effect is as though the user had just logged in and been assigned a fresh job.

RESET is most useful when, through program malfunction or other circumstances, the user becomes confused over what system resources are currently assigned to him. RESET

allows the user to start afresh with a guaranteed clean slate without going through the LOGOUT-LOGIN sequence.

Errors: None. One can always RESET.

The following two commands provide a minimum debugging facility for machine language programs. Each assumes that a program has been placed in core through the Loader or GET, MERGE, or RUN. The program can be freshly loaded or may have been run and interrupted.

12. Examine Location

To examine the contents of a core memory location:

@octal address/

TENEX will type the contents of the word selected as a pair of unsigned octal integers in the format left half,,right half. Unlike other TENEX commands, this one lacks an initial keyword. Also, it requires no confirmation; TENEX types the requested information immediately upon receiving the "/" character.

Examples:

```
@123/              777777,,777777
@100000/           1,,1
```

Errors: NO PROGRAM
NO SUCH PAGE
CAN'T READ THAT PAGE

13. Deposit into Location

To put information into a memory location:

@octal address\octal number<ESC>

or

@octal address\octal number,,octal number<ESC>

The octal number(s) is put into the address selected. Note that the back slash (\) distinguishes Deposit from Examine. Also, with Deposit, the user must confirm with carriage return. If the Deposit changes the status of the page containing the location specified, TENEX will type either [NEW] or [SHARED] to make the user aware that he is either creating a page or making a private copy of a previously

shared page.

The number to be deposited can be written as a single, signed octal integer from Ø through 777777777777 (i.e., occupying up to the 36-bit length of a PDP-1Ø word). Alternatively, the user can specify a pair of unsigned octal integers to occupy the left and right halves of the word selected. In this case the integers can range from Ø through 777777 (up to 18 bits). If two numbers are given, the user can separate them with a pair of commas, as shown above, or with a single comma, space, or altmode. The following examples are all equivalent; in each case location 123(8) would end up containing:

```
ØØØØØ1ØØØØØ7

@123\1ØØØØØ7%
@123\1,,7%
@123\1,7%
@123\1 7%
@123\1$ 7%
@123\-777776777771%
```

Errors: NO PROGRAM
NO SUCH PAGE
CAN'T WRITE THAT PAGE

14. DDT

In addition to the two debugging commands just described, TENEX has a powerful interactive debugging language, DDT. This language is implemented in a program called UDDT, which occupies memory locations 77ØØØØ(8) to 777777(8) of the program being debugged. UDDT is very similar to 1Ø/5Ø DDT, a program written by DEC in conjunction with their 1Ø/5Ø timesharing system, and documented in a DEC manual entitled ASSEMBLY LANGUAGE HANDBOOK. To learn to use UDDT, refer to that manual along with the TENEX USER'S GUIDE which describes the differences between 1Ø/5Ø DDT and UDDT.

To transfer control to UDDT:

```
@DDT%
```

If UDDT is not already loaded in memory with the user's program, the DDT command loads it and establishes communication to the program's symbol table as left in memory by LOADER. If no symbol table is found, TENEX provides one containing JSYS operation codes and other useful symbols. This version of DDT is called SDDT.

Errors: none

15. IDDT

IDDT is an extension of UDDT. It is called "invisible" because it runs in an address space different from that of the program being debugged. Thus it, and the user's symbol table are protected from malfunctions in the user's program. In addition many new commands are available, including those for handling files.

A particularly useful feature of IDDT is the ability to interrupt the program being debugged by typing RUBOUT on the terminal. This causes the user's program to be frozen and control transferred directly to IDDT without going through the EXEC.

IDDT is called by the EXEC command:

```
@IDDT%
```

This may be done after the program has been running for a while. It will preserve the complete state of the user's program, including its fork structure.

IDDT is documented in the TENEX USER'S GUIDE.

16. NO IDDT

It is sometimes desirable to remove the fork containing IDDT from the user's structure so that the program may be resumed running under the EXEC. The command:

```
@NO IDDT
```

will unsplice the IDDT fork.

17. FORK

If the EXEC has more than one fork running under it, its attention may be directed to any one of these using the FORK command. This situation typically occurs after using the IDDT command -- the EXEC will then know about the fork contain the user's program and also the IDDT fork.

```
@FORK 2
```

will direct the EXEC to relative fork 2 so that MEMSTAT, RUNSTAT, etc. may be executed.

Errors: NO SUCH FORK

18. EXEC

Frequently the user wishes to preserve the state of his computation and temporarily invoke a completely new EXEC with no inferior(s). This is done by returning to the EXEC with a "QUIT" command (;H to TECO or IDDT, ↑C to the current EXEC) and executing the EXEC command "EXEC".

The effect will be almost the same as having detached or logged in again, but much faster. Also, this EXEC will be running in the same Job and thus will have access to job-private quantities. EXEC commands may be nested to any reasonable depth.

```
@EXEC↵

 BBN-TENEX 1.31.5, BBN-SYSTEM-A EXEC 1.5Ø.34
@
```

19. QUIT

In order to return to a superior EXEC from one created with the EXEC command:

```
@QUIT↵
@
```

The second "@" was actually typed by the superior EXEC after it has disposed of the inferior and its forks, etc. The CONTINUE command will now resume the original program running under the superior EXEC.

In the above discussion it was assumed that the QUIT command would return control to a superior instance of the EXEC. Frequently the superior fork may contain a different subsystem such as LISP, which itself is inferior to an EXEC. One can usually assume that the top-level fork of his job contains the EXEC.

Errors: NOT LEGAL IN TOP-LEVEL EXEC

2Ø. EDIT

EDIT is an EXEC command which takes a TENEX file name for an argument. Its basic function is to RUN the TECO text editor subsystem, starting it at a special entry point so that it reads in the file named in the EDIT command.

EDIT will accept either an "old" file name or the name of a file that does not yet exist. In the first case the highest numbered version will be read by TECO. In the second case, a null file will be created and TECO's text buffer will be empty.

The name and extension of the EDIT file are remembered by the EXEC so that subsequent uses of the EDIT command can omit the file name all together. In this case the highest version of the named file is selected.

```
@EDIT (FILE) ALPH$A.MAC;31 [Old Version]%
```

Assuming that the file has been edited and a new version written, the command

```
@ED%
```

would transfer to TECO causing it to read ALPHA.MAC;32. If the user now wishes to create a file with the EDIT command, the dialogue would look like:

```
@ED BETA$.MAC;1 [New File]%
```

Note that the altmode caused the extension to be defaulted to .MAC, which is the extension being remembered by the EXEC from the previous EDIT command.

Errors: TECO NOT AVAILABLE
EDIT FILE HAS BEEN DELETED

B. FUTURE ADDITIONS

1. Environment (.ENV) Files and the DUMP Command

Environment files will give the user a mechanism for saving the entire state of his job for continuation in the same or a later session. Environment files will contain 1) the contents of any core memory currently assigned, 2) an indication of the subsystem (if any) currently in use, 3) any information held for the user by the monitor (for example, temporary storage associated with a subsystem or

the active registers of a program), 4) the status of any currently open file (the current location in the file, whether file was open for reading or writing, etc.).

When implemented, TENEX will create Environment files through the DUMP command. GET will operate on Environment as well as Save Core files. With an Environment file, it will restore the user's job to its state at the time the DUMP took place. Also , RUN, START and REENTER will operate on Environment files. If the file contained a machine language program, these commands will start execution in the same manner as with Save Core files. TENEX subsystems also have starting and reentry locations permanently built in. If a subsystem was in use at the time of the DUMP, RUN START, or REENTER will start its execution at the appropriate address.

It will be possible to interrupt a running job by ↑C prior to creating an Environment file. In this case, once the file has been reloaded (for example by GET), CONTINUE will restore job execution as though the interrupt had never happened.

VIII. QUERIES

The TENEX commands in this section answer a user's queries about his job and the system resources he has in use, and also provide general information about the TENEX Executive language and the status of the TENEX system. As noted, certain of these commands can be given prior to logging into the system.

Several commands (MEMSTAT, PISTAT, ↑T, etc.) may be switched among any of the user's forks (normally there is only one) using the FORK command. See VII-A-17.

1. The ? Feature

In entering a TENEX command, the user can type "?" at any point where TENEX appears to expect a keyword or argument. TENEX will respond with a list of allowable keywords or description of the argument(s) expected. Typing "?" instead of an initial keyword will yield a list of all TENEX commands. After an "?" typed within a command, TENEX will retype the command up to that point on the next line, and then pause for the user to make an allowable entry.

Examples:

```
@?
COMMANDS ARE:
 ACCOUNT
 APPEND
   .
   .
   .
 USESTAT
 VERSION
 WHERE

@AVAILABLE ? ONE OF THE FOLLOWING:
 DEVICES
 LINES

@GET ? FILE NAME
 GET
```

2. AVAILABLE LINES/DEVICES

To obtain a list of available terminal input lines (actually jacks on the TENEX input patch panel) or input/output devices:

```
@AVAILABLE LINES/DEVICES
```

Confirmation by CR is not required. The second field defaults to LINES. The user need not be logged in.

Examples:

```
@AVAILABLE LINES
Ø, 1, 2, 3, 4, 5, 6, 7, 1Ø, 11, 12, 13, 14, 15, 16, 17
2Ø, 23, 27, 3Ø, 31, 33, 34, 36, 1Ø1, 1Ø2, 1Ø5

@AVAILABLE DEVICES
MTAØ, MTA1, MTA2, MTA3, DTAØ, DTA1, DTA2, DTA3,
LPT, PTR, PTP, PLT
```

The device codes above refer to magnetic tape units Ø through 3, DECtapes Ø through 3, the lineprinter, paper tape reader and punch, and the plotter.

3. DAYTIME

To obtain the date and time:

```
@DAYTIME
```

Neither confirmation nor login status is required.

Example:

```
@DAYTIME
FRIDAY, APRIL 13, 1973 12:34:56-EST
```

4. WHERE (IS USER)

To find out the terminal line number, job number, and subsystem in use by a particular user:

```
@WHERE (IS USER) user name
```

TENEX will do recognition on the user name. Neither login nor confirmation is required. If the user in question is not logged in, an appropriate reply is given. If the user has one or more detached jobs, TENEX will type "DETACHED" along with any active line numbers. If the user in question is using TENEX via the ARPANET, his remote site will be

included in the information.

Examples:

```
@WHERE (IS USER) SMITH
 TTY21, JOB 19, TELNET
@

@WHERE (IS USER) JONES
 NOT LOGGED IN
@


@WHER FOO%
 TTY105, JOB 47, AMES-TIP, LISP
 DETACHED, JOB 19
@
```

5. SYSTAT

To find out how long the system has been up, who is on and what they are up to:

```
@SYSTAT
```

Neither login nor confirmation is required. SYSTAT types the elapsed time since the system was last brought up, the number of jobs currently active and a table giving the status of each active job. The job table gives the job number, terminal line number (or "DET" if the job is detached), the user name (or "NOT LOGGED IN") and the subsystem in use.

Example:

```
@SYSTAT

 UP 132:37:51   12 JOBS
 LOAD AV   4.51   3.34   2.34

 JOB TTY  USER       SUBSYS
 1   DET  SYSTEM     NETSER
 2   DET  SYSTEM     RSSER
 3   DET  SYSTEM     TIPSER
 4   12   NOT LOGGED IN EXEC
 5   23   PLUMMER    TECO
 6   54   BTHOMAS    EXEC
 7   26   NCC        (PRIV)
 8   46   LEAVITT    RUNOFF
 9   11   IMP        (PRIV)
 11  56   TOMLINSON  EXEC
 14  27   STROLLO    EXEC
```

```
15  6Ø   CHIPMAN   WATCH
```

In the above "EXEC" means that a user is in communication with the TENEX Executive; "(PRIV)" designates user running a private program. Line 12 has made contact with TENEX, but not yet logged in.

The load average numbers typed in the SYSTAT table are useful in determining how busy the computer is. Roughly speaking, these numbers are computed by averaging the number of processors demanded. The three numbers differ in the time constant used in the exponential running sum -- the first is a 1 minute time constant, the second, 5 minutes, and the third has a 15 minute time constant. Thus, the first number gives a rough indication of the short, impulsive demands for computation, while the 15 minute load average tends to show the long term, continuous demand.

SYSTAT will also display a notice of any pending shutdown. The format of this message is the same as that which is printed by LOGIN.

6. VERSION

This command prints the TENEX System and Executive version numbers and date - the same information and format as is typed when the user first contacts TENEX with ↑C. This information should be included in all software trouble reports.

Example:

```
@VERSION%
 BBN-TENEX 1.31.5, BBN-SYSTEM-A EXEC 1.5Ø.34
```

The following commands yield information concerning the user's job, computer usage, and files.

7. JOBSTAT

To find out the TSS Job # assigned, the name of the logged-in user, and the terminal line:

```
@JOBSTAT
```

No confirmation is required.

Example:

```
@JOBSTAT
 TSS JOB 3, USER MYER, TTY25
```

8. RUNSTAT

To determine the status of a job that appears to have stopped running for some reason:

```
@RUNSTAT
```

No confirmation is required. TENEX will respond with a message indicating why the subsystem or program (if any) has stopped. Possible messages include:

```
NO PROGRAM
NEVER STARTED
INTERRUPTED FROM RUNNING AT pc
INTERRUPTED FROM IO WAIT AT pc
HALT AT pc
HALT: ILLEG inst AT pc (or other error condition)
INTERRUPTED FROM FORK WAIT AT pc
INTERRUPTED FROM SLEEP AT pc
BREAKPOINT AT pc
```

In the above, "pc" stands for the current contents of the program counter, i.e., the location at which the interruption, halt, or error occurred.

Examples:

```
@RUNSTAT
 INTERRUPTED FROM IO WAIT AT 7Ø122Ø
 LOAD AV.  = 16.Ø7
```

Here the program was interrupted (by ↑C to the EXEC or Rubout to IDDT) while waiting for input or output at location 7Ø122Ø. The load average is the same as the one-minute load average typed by the SYSTAT command.

```
@RUNSTAT
 NO PROGRAM
```

No program or subsystem had been loaded.

9. FILSTAT

To find out what directory one is connected to, what files are open, and why, and what input/output devices one has assigned:

@FILSTAT

No confirmation is required. TENEX types the connected directory name (if other than the user's primary directory), a table of open files, and a list of assigned input/output devices.

For each file, the open file table lists the Job File Number (JFN - see IV-E-2), the file designator, the kind of access the file is open for, and the current access condition. Access types include:

NOT OPENED
READ
WRITE
EXECUTE
APPEND
PER PAGE TABLE

Access conditions include:

DATA ERROR
EOF (end of file)

For a detailed treatment of file access, see the TENEX JSYS Manual.

Where appropriate, the file byte pointer will be printed for each file listed in the FILSTAT. Note that is is a decimal number and that it is meaningful only for files which are being accessed byte at a time.

Example:

@FILSTAT

```
CONNECTED TO <JONES>.  JFNS:
4    ZILCH.MAC;4                    READ, 128.
3    <SUBSYS>PA1Ø5Ø.SAV;115         READ, EXECUTE
2    <SUBSYS>MACRO.SAV;47           READ, EXECUTE
1    <SYSTEM>EXEC.SAV;15Ø34         READ, EXECUTE
Ø    <PMFDIRØ>JOBPMF.;1ØØØ13;T      READ, WRITE, EXECUTE

DEVICES ASSIGNED TO THIS JOB: MTA1, DTA3
```

In this example the file ZILCH.MAC;4 has had 128 (decimal) bytes read from it. In this case, it was 36-bit bytes which were read on MACRO's behalf by the Compatibility package.

1Ø. MEMSTAT

This command types a report on any core memory assigned to the user. If none is assigned, MEMSTAT types "NO PROGRAM". Otherwise, it types the number of assigned pages, the location and length of the entry vector, if any, and a memory map. The memory map gives the page number(s) for each assigned page or contiguous group of pages. Pages private to the user are designated "PRIVATE". For shared pages, MEMSTAT types the name of the file or number of the fork that "owns" the page, and the local number of the page in question within the owning file or fork. Indirect page pointers are designated by "@".* For most pages MEMSTAT types the type of access allowed: R=read, W=write, E=execute, CW="copy-on-write".

Example:

```
@MEMSTAT

 21 PAGES, ENTRY VECTOR LOC 74ØØØØ LEN 3

737         PRIVATE R, W, E
74Ø-747     <SUBSYS>IDDT.SAV;282 2-11 R, CW, E
75Ø         PRIVATE R, W, E
751         @ FORK 1 5 R, CW, E
752-755     <SUBSYS>IDDT.SAV;282 14-17 R, CW, E
77Ø-774     <SUBSYS>UDDT.SAV;7Ø R, E
775         PRIVATE R, W, E
```

11. USESTAT

To find out how much CPU and console time one has used:

```
@USESTAT
```

No confirmation required.

Example:

```
@USESTAT
 USED Ø:ØØ:17 IN 1:12:11
```

Seventeen seconds of CPU time were used during 1 hour, 12 minutes, 11 seconds of connected time.

---

*Shared pages, forks, ownership, and indirect page pointers are discussed in detail in various TENEX documents. For most users, the information tabulated concerning private pages will suffice.

12. DSKSTAT

Types the number of disk pages occupied by all files in the currently connected directory, sorted according to number of deleted and undeleted pages. The allocation associated with the connected directory and the system total statistics are also printed.

Example:

```
@DSKSTAT
327 TOTAL PAGES IN USE - 250 ALLOWED, 299 UNDELETED, 28 DELETED
SYSTEM TOTAL: 2608 PAGES LEFT, 88517 USED
```

13. PISTAT

Types information regarding the state of the Pseudo-interrupt system for the user fork. This includes whether the PSI system is turned on, the locations of the level table and channel table, the channel mask and the breaks-in-progress word.

Example:

```
@PISTAT
PSI IS ON, LEVTAB=1022, CHNTAB=1025, CHN MASK=400000200000,
BIP=0
```

14. FORKSTAT

Types the fork structure below the EXEC. The status of each fork is also typed. Forks for which the EXEC has not assigned a relative fork handle are indicated with "**" instead of the fork number.

```
@FORKSTAT%

FORK 1:   BREAKPOINT AT 105
   FORK 6:   INTERRUPTED FROM RUNNING AT 40103
   FORK **:   INTERRUPTED FROM IO WAIT AT 1002
FORK 2:   INTERRUPTED FROM IO WAIT AT 747641
FORK 5:   NO PROGRAM
```

15. Control-T

Frequently the user would like to know whether his program is making progress. To facilitate this, the EXEC arms T (control-T) as an interrupt character, which causes a RUNSTAT and USESTAT to be performed. Note that the user's program is not stopped while this information is being typed

and if his program is typing out, ↑T will result in a garbled typescript.

Example of ↑T interrupt typeout:

```
IO WAIT AT 4263
LOAD AV.  = 14.85, USED Ø:Ø1:46.2 IN 3:39:3Ø
```

IX. TERMINAL CHARACTERISTICS COMMANDS

These commands allow the user to communicate to TENEX the salient characteristics of his terminal.

1. Half or Full Duplex

To inform TENEX whether one's terminal is half or full duplex:

@HALFDUPLEX%
OR
@FULLDUPLEX%

For an explanation of half vs. full duplex terminals, see III-B-1. These commands can be given prior to logging in. Unless informed otherwise, TENEX assumes full duplex terminals.

2. Other Terminal Features

If one's terminal has built-in mechanical tab stops, a form-feed mechanism or lower case characters, these facts can be communicated to TENEX by:

@TABS%
@FORMFEED%
@LOWERCASE%

These commands can be negated by preceding them with the keyword NO:

@NO TABS%
@NO FORMFEED%
@NO LOWERCASE%

Until informed otherwise, TENEX assumes the absence of tabs and formfeed, and the presence of lower case. With the above commands as well as HALFDUPLEX and FULLDUPLEX altmode is acceptable for termination.

3. STOPS

To set software tab stops for terminals lacking a tab mechanism:

@STOPS n, n . . . n %

This command takes a list of decimal numbers (between 1 and 1Ø7 inclusive), separated by commas. The numbers specify character positions for software tab stops. If no STOPS command is given, tab stops are every eight columns.

Example:

@STOPS 1Ø,23,47,8Ø%

Errors: "?" for argument greater than 1Ø7

The commands above control output from TENEX to the user's terminal. For example, TABS and FORMFEED will cause TENEX to output these characters while NO TABS or NO FORMFEED will cause TENEX to simulate their action with spaces or linefeeds, respectively. LOWERCASE permits TENEX to transmit lower case characters to a terminal, while NO LOWERCASE forces a conversion to upper case prior to transmission. The RAISE command below also affects case conversion, but applies to input from a terminal, rather than output to it.

4. RAISE

With a dual case terminal one can type in lower case but have characters be received (and echoed) in upper case by:

@RAISE%

The command:

@NO RAISE%

turns off RAISE; characters are received and echoed as typed.

5. INDICATE

To change the effect of form feed ((CTRL)-L) to merely print "↑L" instead of advancing the paper to the top of the next page:

@INDICATE (FORM FEED)%

Some users prefer this mode to eliminate the blank paper spewed out of the terminal when running certain subsystems or making a listing (above). To revert to feeding paper on form feeds, use FORMFEED or NO FORMFEED.

6. Padding

Several different technologies are in use for the manufacture of computer terminals -- mechanical, thermal, CRT, etc. Each of these has its own peculiarities, but the main difference is in the amount and kind (return, linefeed, formfeed, tab) of "padding" required. The command:

@TERMINAL (TYPE IS) terminal type

allows the user to inform TENEX of his terminal's mechanical characteristics. The current list of terminal types includes 33, 35, 37, AJ, ANDERSON-JACOBSON, COMPUTER-DEVICES, EXECUPORT, LA3Ø, NCR, NVT, TERMINET, and TI. The current list of available terminal types may be typed by answering "?" to the TERMINAL (TYPE IS) command, instead of a valid type. New terminal types are added to TENEX as the need arises.

Example:s

@TERMINAL (TYPE IS) EXECUPORT%

Or,

@TE TI%

TENEX assumes TERMINAL TYPE 33 unless some other is specified.

7. WIDTH

TENEX automatically "folds" a typed line if it is longer than 72 columns. This parameter is changeable using the WIDTH command.

@WIDTH (IS) 8Ø%

The previous command set the number of columns to 8Ø. A special case is "WIDTH Ø" which causes TENEX to turn off the line folding feature.

X. TERMINAL LINKING AND ADVISING

This section describes commands for dealing with the LINK and ADVISE features of TENEX, which allow users to communicate on-line.

1. LINK

The LINK command causes each of two users to be able to see output which is being typed on the other's terminal. LINK is implemented by mixing output streams. That is, characters output by programs running on either job, or echoes for characters typed on either of the terminals are directed to the printers on both terminals.

Several features are to be noted regarding LINKs:

(1) Passwords are secure during LINKs since echoing is turned off during password input.

(2) Characters typed by one party are never seen by programs on the other's end. The echoes are simply printed on his end.

(3) Characters typed by each person continue to be seen by programs running on his own end, just as if no LINK existed.

(4) Multiple LINKS are permitted, but users must keep the implementation details in mind. Thus, if A is LINKed to B, and B is LINKed to C, A will see output produced by his own typing as well as B's, but not C's. B will see output produced by all three. C will see only that due to himself and B, but not A.

The general form of a LINK command is:

@LINK (TO) terminal number or user name

Recognition will be performed on user names. LINK requires confirmation, but not login. The "automatic logout" mechanism is still in effect for users who are not logged in -- even if they establish a LINK.

If the LINK command is successful, the LINK will be announced by a message typed on both terminals. If the LINK is REFUSEd (see section 3), TENEX will use the terminal bell much like a telephone bell to notify the object terminal of

the LINK attempt. The object terminal may chose to ignore the LINK attempt or may RECEIVE (see section 4) the LINK.

In the case where the user being linked to has more than one non-DETACHed job, TENEX will list these jobs and ask for a specific terminal number. The user may either type the number or default it to the last entry in the list by typing a carriage return.

Example:

```
@LINK (TO) CHIPMAN%
 TTY1Ø0, EXEC
 TTY27, RUNOFF
 TTY34, SYSDPY
TTY:   1Ø0%

 LINK FROM PLUMMER, TTY57
@;HI STEVE.  WOULD YOU PLEASE MOUNT MY DECTAPE 42?
@;SURE....ITS ON DTA2:
@;THANKS, BYE
@BREAK (LINKS)
```

In this example user CHIPMAN had three jobs which were listed by the LINK command. The job to which terminal 1Ø0 was attached was selected because it was in the EXEC. Note that the LINK FROM ... message was typed because the LINK was not being REFUSEd, and the conversation proceeded. Since both parties were in the EXEC, each line was begun with a semicolon so that it would be considered as a comment and not an EXEC command.

Had user PLUMMER known that CHIPMAN was available on line 1Ø0, he could have shortened the process by typing:

```
@LINK 1Ø0%
```

That is, the terminal number was a complete identification.

Errors: NOT LOGGED IN
REFUSED
NON-EXISTENT TERMINAL NUMBER

2. ADVISE

ADVISE is similar to LINK except that the input stream to the advised job is formed by merging the characters typed on both parties' keyboards. Characters typed of the advisor's keyboard do not go to the advisor's job.

The general form and operation of the ADVISE command are the same as LINK, described above. ADVISE takes either a user name or a terminal number as the argument.

The normal state of a job is that it is refusing advice. In order to allow someone to use the advice mechanism, one must first execute a RECEIVE ADVICE command (see below). The people involved usually arrange matters using a terminal LINK beforehand. Executing an ADVISE command will fail if the person being ADVISEd has not executed a RECEIVE ADVISE command, specifying the advisor's terminal number.

While advising another terminal, a control-C will automatically break the ADVISE connection, but the output link remains in effect. If the advisor wishes to cause a ↑C to be typed in on the advisee's terminal, it is done by typing a control-Y.

Errors: REFUSED
NOT LOGGED IN
NON-EXISTENT TERMINAL NUMBER
TERMINAL HAS ADVICE IN PROGRESS

3. BREAK

When a conversation carried on over a terminal LINK or ADVISE is to be ended, the command is:

@BREAK (LINKS)%

This breaks all links to the terminal which executes the BREAK. BREAK also implies REFUSE ADVISE.

4. REFUSE

If one is outputting a document on his terminal or is TYPEing a file, the message announcing a successful LINK would destroy the listing. This can be avoided by:

@REFUSE (LINKS)%

The REFUSE command may be given after a LINK or ADVISE connection has been established. This will prevent more LINKs from being received, avoiding problems with who typed what.

5. RECEIVE

RECEIVE is the opposite of REFUSE. It specifies that advice will be accepted at the instant the command is issued or that LINK attempts will be successful from then on. If no keywords are given after the command, RECEIVE LINKS will be assumed. In the case of RECEIVE ADVICE, a terminal number must be supplied.

Examples:

```
@RECEIVE%
```

Or,

```
@RECEIVE ADVICE (FROM) 14%
```

Unless told otherwise, TENEX will assume that you are willing to RECEIVE LINKs.

XI. ARPANET COMMANDS

The TENEX comands in this section deal with the ARPANET; they are not implemented at TENEX installations which are not connected to the Network.

1. NETLOAD

To obtain an indication of which sites are operational, their five minute load averages and the number of users at each site:

```
@NETLOAD
 SITE            LOAD  USERS
SRI-ARC          6.67   24
USC-ISI            ?     ?
UTAH-1Ø          4.Ø7   17
BBN-TENEX        9.63   37
BBN-TENEXB       Ø.57    2
CASE-1Ø          1.88    9
```

In this example, the site USC-ISI was in the progress of resuming operation after a service interruption. Numbers will replace the question marks a few minutes after the appropriate "handshaking" has taken place.

Errors:

NETWORK LOAD STATISTICS NOT AVAILABLE
SERVER DEAD

The first of these means that the system file in which the statistics are kept, has been temporarily made inaccessible. The SERVER DEAD error results when the statistics gathering program has been taken out of operation.

NETLOAD does not require confirmation or login.

2. Future Additions

As various issues such as Network-wide naming of files and users, password security, etc., are resolved, the EXEC will be extended, allowing users to take full advantage of the facilities provided by the ARPANET.

XII. FUTURE COMMAND GROUPS

1. Input/Output Redirection

Normally, programs and subsystems accept commands and parameters from and send messages to the user's terminal. Commands in this section will switch such input and output to other devices or files.

$ 9, 11

% 11

& 16

(CTRL)-A 17
(CTRL)-C 6, 67, 1Ø6
(CTRL)-F 33, 38, 4Ø
(CTRL)-L 16, 64
(CTRL)-R 18
(CTRL)-T 99
(CTRL)-W 18
(CTRL)-X 18
(CTRL)-Y 1Ø6
(CTRL)-Z 57, 58

* 47, 54

.BAS 35
.F4 35
.MAC 4, 35
.REL 4, 35
.SAV 4, 35

1Ø/5Ø DDT 9Ø

36 bit bytes 54

7 bit bytes 54

8 bit bytes 54

; 7, 16
;A 42
;P 42
;T 42

? 18, 92

@ 6, 7
@@ 15, 49, 55

Abbreviated Input 9
ACCESS COMMANDS 23
ACCOUNT 22, 41, 45, 49, 59
ALGOL 74
Allotment number 36
ALPHABETIC 5Ø
Altmode 9, 12, 33, 38, 4Ø
Analog to digital converter . 32
APPEND 57

Append protection 43
ARCHIVE 66
Archive status 67
Archive-pending file 60
Argument 7
Argument Input 10
ARPANET 108
ASCII 54, 55
Assembler 4, 74
ASSIGN 3, 46, 70
ATTACH 25, 28
AUTHOR 49
Autologout 25, 104
AVAILABLE LINES/DEVICES . . . 93

BASIC 74
BCDTAP 75
BCPL 74
BEGIN 51
Bell ringing 104
BINARY 55
Binary 4
Binary data 54
BINCOM 75
BLISS 74
BREAK 106
BSYS 74
Byte size 54, 58
Byte size record 54
Bytes 49, 54

Carriage Return 10, 12
CCL 74
CHANGE 22, 26
Checksum 56
CHESS 76
CHONOLOGICAL 50
CLEAR 61
Clearing jobs 85
COBOL 74
Comma 11, 15, 40, 49, 55
Command Components 7
Command Field 8
Command Input and Recognition . 8
Command line 67
Command Terminators 10
Commands from file 2, 109
Comments 7, 16
Compatibility 97
Confirmation 67
Confirming Carriage Return . . 10
CONNECT 33, 62
Continuation 16

CONTINUE 27, 85
CONVERT 75
COPY 3, 52
Copy on write 98
COPY subcommands 54
COPYM 75
CRAM 5Ø
CREF 75

Date 49, 51, 93
Daytime 93
DDT 75, 87
DDT-1Ø 9Ø
DEASSIGN 3, 7Ø, 85
Debugging 75, 86
DECtape 3Ø, 54, 69
DECtape units 32
Default Command 13
Default file protection . . . 45
Default Values 12, 37, 4Ø
DEFERRED 66
DEFINE 68, 72
DELETE 5, 59, 66
Deleted file 61
DELETED FILES ONLY 51
DELVER 75
Deposit into location 86
DETACH 26, 65
Device 3Ø, 31, 54, 93
Device codes 31
Device handling 69
Digital to analog converter . 32
DIRECTORY 3, 5, 48
Directory groups 44, 63
Directory name 3Ø, 32
Directory protection 43
Disk 3Ø, 32, 54
Disk allocation 24, 25, 63
Disk file structure 56, 58
Disk packs 69
Disposal 32
DO 75
DOCTOR 76
DON'T ARCHIVE 66
DON'T DELETE 66
DOUBLESPACE 5Ø
DSKAGE 75
DSKSTAT 99
DTACOPY 75
DUMP 9Ø
DUMPER 75

ECAP 76

EDIT 90
Editing and Errors in TENEX Commands 17
End of file pointer 57
Entry vector 77, 80, 98
Environment files 90
EOF 57
EPHEMERAL 67
Errors 17, 18
EVERYTHING 50
Examine location 86
EXEC 89
Execute protection 43
Executive Language Structure . 6
EXPUNGE 61
Extension 30, 34

FAIL 74
Field value 8, 12
Fields 8
FILCOM 75
FILE BYTE POINTER 97
File byte pointer 97
File commands 47
File descriptors 30
File designator errors 46
File designator lists 40, 53
File designators 30
File directory 69
File groups 47
File name 30, 33
File protection 42, 47
File protection field 43
File protection number 43
File sharing group 43
File system 1, 30
FILSTAT 96
FIOCNV 75
FLIST 75
FLOW 76
FORK 88, 92
Fork status 99
Fork structure 99
Forks 98
FORKSTAT 99
Format Conventions 20
Format Options 15
FORMFEED 16, 64, 65, 66, 101, 102
FORTRAN IV 74
FRKCOM 75
FTP 76
FUDGE2 74
Full Input 8
FULLDUPLEX 21, 101

GET 77, 81, 90
Gibberish characters 22
GLOB 75
GOTO 77, 84
GRIPE 76
Group access 63
Group numbers 44
Group protection 43
Groups 43, 44
GTJFN 67

HALFDUPLEX 22, 101
Harmless commands 11
HEADING 64
Holes 56, 58

I/O devices 96
IDDT 75, 88
IMAGE 55
IMAGE BINARY 56
IMGPTP.SAV 76
IMMEDIATE 67
INDICATE 102
Indicate nulls 65
inferior exec 89
Inferior EXEC 89
INFILE 27
Input/output devices 30
Input/output redirection . . . 109
INTERROGATE 67
Interruption 96
Invisible DDT 75, 88

JFN 46, 96
JFNCLOSE 46, 62
Job file number 46
Job storage block 46
Jobs 22
JOBSTAT 95
JOTTO 76
JSB 46
JSYS 1

Keyword 7

LBLOCK 76
LENGTH 49, 65
LIFE 76
Line folding 103
Line printer 32, 54, 69
Lines 93
LINK 104

Linking loader 74
LISP 74
LIST 3, 63
Load average 95, 96, 108
LOADER 4, 74, 87
Locate file 67
Logical names 72
LOGIN 2, 23
LOGIN message 23
LOGOUT 5, 25, 65
LOGOUT message 25
LOWERCASE 16, 101, 102
LPT 51
LPTPLT 76

MACRO 4, 74
Magnetic tape 30, 54, 69
Magnetic tape units 32
MAIL 24, 67
MAILER 76
Maintainence programs 74
Memory access 98
Memory map 98
Memory sharing 79
MEMSTAT 98
MERGE 77, 82, 90
Merging files 82
Message 23, 24
MIDAS 74
MINCOP 75
MOUNT 46, 70
mountable disk packs 69
Mountable disk packs 32
Moving information 30
MTACPY 75
Multiple file designators . . 40, 53, 58

NETLOAD 108
NETSTAT 76
Network 108
NIL: 31
NIN 67
NO 64, 101
NO (HEADING) 50
NO IDDT 88
Noise word 7
NOT EPHEMERAL 68
Null extension 34, 38
Null Values 12, 14
Nulls in files 65

Open file 46, 96
Optional disk file descriptors 41

Other protection 43
OUTFILE 27
OUTPUT 65
Output buffer 6
OUTPUT TO FILE 51

Packing 54
Page numbers 80
Pages 49, 64, 82
PALX 74
Paper tape 30, 55
Paper tape punch 32, 69
Paper tape reader 32, 69
Passwords 21
PAUSE 66
PDP-10 computer 1
PISTAT 99
Plotter 32
PPL 74
Primary directory 32
private disk packs 69
Private disk packs 32
Private mail 24
Private pages 98
Program Control 77
Program interruption 84, 85
PROTECTION 49
Protection status 41

QFD 52
QUERIES 25, 92
QUIT 89

Range of version numbers . . . 35
Read protection 43, 47
READMAIL 24, 76
RECEIVE 105
RECEIVE ADVICE 107
RECEIVE LINKS 107
Recognition Input 9
Recognition of file descriptors 38
REENTER 27, 77, 84
REFUSE 106
Relevant descriptors 37
RELRIM 75
RENAME 58
Reserving devices 69
RESET 67, 85
Retrieve from archive 67
REVERSE 50
REWIND 71
RJS 76
RSEXEC 76

Rubout 18
RUN 13, 77, 83, 90
RUNFIL 74
RUNOFF 74
RUNSTAT 96

SAIL 74
SAVE 4, 77, 78
Save Core Files 56, 58, 77
Scope terminal 22
Scratch file 61
SDDT 75
Self protection 43
Semicolon 7
SEPARATE LINES 50
Shared pages 79, 98
SHUT 46, 62
Signaling TENEX 6
Sites 108
SIZE 49
SNDMSG 24, 76
SNOBOL 74
SNOW 76
SORT 74
Space 9, 16
SPACING 64
Specific terminal lines . . . 32
spooler 64
SRCCOM 75
SSAVE 56, 77, 79
Standard extensions 35
START 27, 77, 84
Starting programs 83, 84, 85
STATUS 67
STOPS 101
Storing information 30
Subcommands 11, 15, 49, 54, 55, 58, 64, 66
SUBSYS 73
Subsystem control 3, 73
Subsystems 1
Synonyms 72
System LOGIN message 23

TABS 16, 25, 101, 102
TAPCVT 76
TECO 3, 74, 90
TELNET 76
Temporary file 41, 42, 61
TEN50 50
TENEX Command Structure . . . 7
TENEX Comment Symbol 7
TENEX Executive Language . . . 1
TENEX JSYS Manual 1

TENEX Ready Symbol 7
Terminal 54, 69
Terminal Characteristics Commands 25, 101
TERMINAL type 103
Text editing 74
Time 93
TIMES 49
Top-level EXEC 89
TSS Job Number 22
TTYTRB 76
TTYTST 76
TYPBIN 75
TYPE 65
Typing file designators . . . 36
TYPREL 75

UDDT 75, 87, 90
Unauthorized access 42
UNDELETE 60
Unit number 31
UNLOAD 71
UNMOUNT 71
Unnamed information 32
Unpacking 54
Upper case 102
Use charges 22
User groups 44
User message 24
User Names 21
User's terminal 32
Users 108
USESTAT 98

Value 8
VERBOSE 50
Version 95
Version default rules 35
Version number 30, 35, 95
Version number 0 36
Virtual computer 1

WATCH 76
WHERE is 93
WIDTH 65, 103
Word size 54
Write protection 43, 47

↑A 17
↑C 6, 67, 106
↑F 33, 38, 40
↑L 16, 64
↑R 18
↑T 99

↑W 18
↑X 18
↑Y 106
↑Z 57, 58

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138
(617) 491-1850