User's Manual

TESS/3000

Text Editing Sub-System

Computer Consultants & Service Center, Inc.
2001 Hanley Center, St. Louis, MO 63144

(PRT 16 APR 85)

LIST OF EFFECTIVE SECTIONS

The List of Effective Sections lists the most recently updated Sections. Within the manual, pages are dated with the "Print Date" at the bottom of the page. Line changes are marked with a vertical bar in the right margin.

Section	Release	Effective Date
2.1	A.09	Nov. 1985
2.6.3	A.09	Nov. 1985
2.6.5	A.09	Nov. 1985
2.8	A.09	Nov. 1985
5.2.6	A.09	Nov. 1985
5.2.7	A.09	Nov. 1985
5.3.21	A.09	Nov. 1985
6.6.2	A.09	Nov. 1985
6.7	A.09	Nov. 1985

TABLE OF CONTENTS

1.0 INTRODUCTION

Originally, all files were maintained using punched cards. This was a very slow process which involved finding the card to be changed, loading it into the keypunch, and then duplicating as much as was possible with the changes keyed in where necessary. Much has changed in the way computers do their job, but unfortunately, the way in which files are maintained has changed little from the basic idea of punched cards. There are essentially three different types of text editors on the market today: *Character-oriented, Line-oriented* and *Screen-oriented* editors.

Character-oriented editors (eg: TECO) are very flexible because they allow the user to position the file pointer (dot) anywhere in the file. That is, not only can the dot be placed at the front of a line, but anywhere within the line as well. This flexibility, however, does not come without a price. Character editors require the user to visualize all movement of the dot and to keep track of its position throughout the editing process. This is no small problem, and can take years to master.

Line-oriented editors (eg: HP EDITOR) lose some of the flexibility awarded with character editors, but they do manage to take much of the burden off of the user. The real drawback of these editors, however, is that they show too much. An advanced user will quickly become impatient waiting for prompts and rewriting of lines after each modification.

Screen-oriented editors throw out the confining concept of a line pointer or dot. They take the view that the Terminal screen is a window into the file. Through this window, one may look into any portion of the file and the cursor can move anywhere within the window. The screen contains no user prompts, only the text in the file is displayed on the screen. This allows the user to view his file as it exists—whole and uninterrupted. The cursor then becomes the means for making modifications; typing over a character changes that character. In other words, the screen is a mirror image of the file.

All editor commands are assigned to *control characters* allowing the editing tasks to be expressed in a concise yet natural way. Learning to use the editor is analogous to learning to touch-type on a regular typewriter. Each of the 32 *control characters* has two possible functions—the normal function and the shifted function. The shifted set is reached by a software shift key that activates the additional 32 commands. Generally, special Terminal function keys are avoided since they cause the user's hands to stray from the typewriter keyboard. However, some function keys may be utilized under certain circumstances.

The cursor is the focus for almost all editor commands. The cursor movement commands allow the cursor to be moved anywhere within the current display window. If the cursor attempts to move beyond a display boundary, the cursor will either do nothing, wrap around, or cause the window to scroll in the direction the cursor was attempting to move, depending on the situation.

Screen-oriented editors have gained wide acceptance from both the academic and industrial fields because of the unique way they combine ease of use with great flexibility. They may be found on all types of machines from the smallest micros to the largest mainframes. TESS now offers HP-3000 users the opportunity to take advantage of this new concept in editing.

1.1 SCREEN EDITING CONCEPT

Screen editors are designed for efficiency and economy; economy of thinking as well as economy of key strokes. In most editors, if there is a mistake, one must back-up and re-key everything from the mistake on. Not only is this method repetitious, but it is also inefficient. The most important concept of screen editing is to preserve the text on the screen and change only what is necessary. For this reason, a great majority of the 64 commands are designed to manipulate the cursor and the screen.

Two modes of editing exist. For adding more than one line, the add mode is used. The add mode is a line mode wherein cursor and screen control is unnecessary. Input is done in the standard left to right progression. Text from cassette tapes, floppy disc, or hard copy may then be input in line editor fashion, using the carriage return to end a line and the ^Y to exit the add mode. For all other editing needs, character mode is used. The character mode is the normal mode of TESS. Any alpha, numeric, special or blank keystroke will be transmitted to the file, while any control character will instruct the program to perform a specific editing function. **Block mode and local cursor editing keys are NOT used.**

The commands are complementary so that one command may undo another, ie. a line may be split into two lines and then re-joined. Mistaken characters may be undone by using the undo command. Using this key, lines may be undone to the way they appeared when they rolled onto the screen previous to modification. Undoing a newly inserted line will erase the line. Additionally, the last 24 deleted lines are stored in last-in first-out order for recovery above the current line denoted by the cursor.

TESS is frugal with system resources. The stack is kept trim, the program segments are small and the algorithms are tuned for CPU and elapsed time economy. Text and keep time, in the worst case, is 18% faster than that obtainable with the HP EDITOR, and at times can be instantaneous. TESS only writes when absolutely necessary. These savings, along with the intrinsic savings of screen editing, provide dramatic increases in productivity and decreases in system overhead.

2.0 GENERAL INFORMATION

2.1 INSTALLATION

The Demonstration package includes the following items:

— 800 or 1600 bpi STORE tape

— TESS User's Manual

— AIDE User's Manual

— TESS and AIDE Tutorials

— These installation instructions.

The tape includes the following files:

CCSCSOFT.PUB.SYS	Job stream for installation.
TESS.PUB.CCSCSOFT	Object code for TESS/3000.
AIDE.PUB.CCSCSOFT	Object code for AIDE/3000.
SL.PUB.CCSCSOFT	Segmented Library for AIDE.
CAPSW.PUB.CCSCSOFT	Utility program for adding PH to copies of BASIC and QUERY in the PUB.CCSCSOFT group.
EXAMPLE.DOC.CCSCSOFT	Used with the TESS and AIDE Tutorials.
INSTALL.DOC.CCSCSOFT	This document.
CMDFILE.DATA.CCSCSOFT	Contains pre-defined Multi-Function strings (macros) which may be loaded for execution.
ENVFILE.DATA.CCSCSOFT	Example of a TESS/3000 'environment file' (in the current release the environment file is used for automatic soft-key downloading).
TESSUDC.DATA.CCSCSOFT	Contains samples of User-Defined Command definitions for invocation of TESS and AIDE.
@.TERMCAP.CCSCSOFT	The TERMCAP Group is the TESS collection of TERMCAP files.

(1) Sign on as SYSTEM MANAGER.

(2) Mount the CCSC installation tape and press 'ON-LINE'.

(3) RESTORE the file CCSCSOFT.PUB.SYS from the installation tape:

```
:FILE CCSCSOFT;DEV=TAPE
:RESTORE *CCSCSOFT;@.PUB.SYS;SHOW
```

CCSCSOFT.PUB.SYS should now exist on your system. the tape should remain mounted.

(4) Temporarily remove the system MANAGER and SYS account passwords and if you have passwords on QUERY or BASIC, they must be altered as well. For example:

```
:ALTUSER;PASS=
:ALTACCT;PASS=
:RENAME BASIC.PUB.SYS/PASS,BASIC.PUB.SYS
:RENAME QUERY.PUB.SYS/PASS,QUERY.PUB.SYS
```

-OR-

Using EDITOR, insert the appropriate passwords into line 1 of CCSCSOFT.PUB.SYS. If there are passwords on QUERY and BASIC, they must be added to the CCSCSOFT job stream on lines 49 and 70.

(5) Stream the job to create and restore the CCSCSOFT account.

```
:STREAM CCSCSOFT.PUB.SYS
```

The tape request 'CCSCSOFT' will appear on the system console. You must issue a reply and again press 'ON-LINE'. Upon job completion, all TESS/AIDE files will be installed.

(6) Replace your system MANGER, SYS account, and PROGRAM passwords.

(7) In order to run TESS/AIDE, the system/user level UDC must be set.

```
:SETCATALOG your.present.udcfile,TESSUDC.DATA.CCSCSOFT;SYSTEM
```

NOTE: You may activate different user-level UDCs for different users in order to employ unique names for the TESS Workfile. The TESS Workfile is automatically retained in the sign-on group at the end of a TESS session unless the name 'FASTEXT' is equated to $NULL. For further information on UDCs, see §3 of the TESS Reference Manual.

(8) Please return the tape to CCSC.

The group and account where TESS resides must have privileged mode (PM), and process handling (PH) capabilities. However, the privileged code may be turned off with the NOPRIV option of the INFO string.

2.2 SUPPORTED TERMINALS

TESS is designed with the HP-26xx series Terminals as the standard. When new HP Terminals are marketed, they are added to the list of supported Terminals. A list of supported Terminals may be found in §6.3.

Further, a wide variety of non-HP Terminals may be used with TESS through the TERMCAP file. If the Terminal has "insert line" and "delete line" capabilities, a TERMCAP file may be used. (See §6.7 TERMCAP files.)

2.3 TERMINAL CONFIGURATION

Terminal soft keys and cursor movement keys may be used under certain circumstances. **Terminal editing keys such as INSERT CHAR and DELETE LINE are not used.** All editing operations are programmatically controlled.

The wrap-around line option must be turned off in order for TESS to function properly. Strap 'C' must either be opened programmatically or opened by hand. If an error 31 is issued while running TESS, the Terminal strapping could not be set automatically and will have to be set by hand. Other configuration should be set to the standard HP operating modes. Block mode must be off. For further assistance on setting straps, see the Terminal's User Manual.

It is recommended that the Terminal have 8K of memory. This is not a requirement, but some function keys that extensively use a *second page* of memory, such as the *help* key, may not fit into less memory. Data on the file will not be effected but the screen will have to be refreshed after a *second page* of memory is used because the *first page* will have been forced out of the Terminal's upper memory. Additionally, it is recommended to run the Terminal at 1200 baud minimum. TESS will work at slower speeds, however, drawing the full screen at 300 baud might appear to take too much time. TESS is not intended to run on MTS lines.

A full list of HP (and HP compatible) Terminals with their configuration requirements is given in the Appendix (§6.3).

2.4 LANGUAGE AND SPECIAL CAPABILITIES

TESS is written in SPL with assembly routines where speed is vital. NOWAIT I/O is used in character mode, making I/O to the Terminal as fast as possible. Actual character key response time, however, depends entirely on the system's I/O load at the time. *Priv mode* is used to open the Terminal for NOWAIT IO (although, this feature may be overridden through the NOPRIV option of the INFO string). *Process Handling* is used for compiling inside of TESS.

2.5 NECESSARY FILES

2.5.1 EDIT FILE

TESS can edit any fixed length ASCII file up through 80 character record lengths (88 numbered). If a *Direct* Terminal is used, the length may be extended to 132 characters (140 numbered). The Limit on the number of records is 30,767. The formal designation is EDITFILE. **This file should be free of control codes.**

WARNING: When editing a numbered file, the file may be renumbered at keep time. Thus, if it is important to keep the numbers intact on a file, **DO NOT** use TESS on that file.

2.6 OPTIONAL FILES

2.6.1 WORK FILE

All work is done on a copy of the edit file, with the changes being made at 'keep' time. This is done so that if an interruption occurs during editing, recovery of the work area is possible. The work file consists of the letter 'E' followed by seven numbers. These numbers are a combination of the date, the hour of the day, and the second of the hour divided by 5. The TESS 'E' file, which is NOT compatible with the HP EDITOR 'K' file, has its own file code of 123. After normal termination of TESS, this file may be automatically kept for fast re-texting, or it may be purged. Equated names other than $NULL will be kept while a Workfile given the name of $NULL will be purged. The formal designation for this file is FASTEXT.

2.6.2 LIST FILE

When a listing is desired, a device needs to be specified through a file equation.
ie. FILE EDITLIST;DEV=LP

2.6.3 ENVIRONMENT FILE—HP-Softkeys and Remap Control Keys

DOWNLOADING SOFTKEYS

An example of the ENVFILE appears in the Appendix. The Environment file (formal file designator ENVFILE) is a data file containing softkey instructions for HP terminals and/or mapping instructions for control keys on most terminals. These two functions are designated within the environment file with **$CONTROL** statements. These features are activated with the file equation that is usually placed in the USER UDC:

FILE ENVFILE=ENVFILENAME.GROUP.ACCOUNT

To automatically set up softkeys with predetermined functions, insert the statement **$CONTROL SOFTKEYS** on a line by itself ahead of the softkey assignments. This signals that the following section defines HP softkeys.

Use of softkeys on non-HP terminals is presently not supported. If an HP terminal is used which supports loadable softkeys and a **$CONTROL SOFTKEYS** section of the enviroment file is succesfully interpreted, the softkeys will be loaded automatically at TESS start.

Commonly used control keys may be downloaded into the softkeys so that the "CTRL" key does not need to be pressed as often. The first eight records correspond to the eight soft keys. If more levels of keys are needed, up to four levels of keys may be included by adding them to the file, always in groups of eight.

When more than one level of softkeys are used, the tilda becomes a special function key that means *next level* in the first tier and *previous level* when used in the second tier. **Once the tilda becomes a function key for switching softkey levels, it will no longer print a tilda on the screen.** Downloading the tilda into *f1* makes *f1* the menu level control key. Softkey *f1* will control the **NEXT/PREV** level of softkeys. By pressing the 2nd tier key and *f1*, the previous level of softkeys will be activated and the appropriate labels displayed. If the tilda character needs to be accessed while softkey levels are active, the Multi-Function key may be set to generate the tilda. (see §5.3.21 Multi-Function).

Two different data elements may appear in softkey loading; A control key command or a literal string. To download a control key, **for use in character mode only**, the accent mark "^" is placed in column one, followed by the ASCII character that represents the desired function. To download a literal string **for use in add mode only** the string should begin and end with a double quote. A common use for the literal string feature is to download two blanks for easy tabbing by twos. If column one is not "^" or " " " or "<" then a literal string is assumed. Terminals with label capability may download labels by enclosing the label with chevrons "<" ">" and placing it ahead of the function or string in column one. For example:

<del char>^D
<TAB X 2sIN ADDMD>"□□"

The first line will download the delete character function and label the key "del char". **While TESS is in character mode, as opposed to add line mode, only one control character may be downloaded into a softkey.** The second line will download two blanks for use in add-mode only. Notice that the labels are split into two lines on the terminal. The split is after 8 characters. Since in add mode there are no command tiers, the entire label may be used to explain the softkey function.

A second tier function requires the use of two user function keys—a second tier shift key and the desired function. In add mode, control keys are invalid but literal strings of characters may be utilized.

Note: Certain characters may not normally be downloaded into the softkeys. Control "E" will not load since the Terminal confuses it with computer handshaking (unless "Enq/Ack" is disabled in the Terminal). Control "@" is a null, and the Terminal ignores all nulls (unless the "Strip Nulls" option is disabled in the Terminal). In addition, on 2626 Terminals, control "N" and "O" are command codes that invoke alternate character sets and these two control keys can not be downloaded. If these functions are necessary, they may be remapped to keys that can be downloaded (see remap section below).

REMAPPING CONTROL KEYS

An example of key remapping appears in the Appendix. To REMAP the keyboard the statement **$CONTROL REMAP** must appear on a line by itself, followed by the remapped keys. Any of the TESS functions can be remapped so that TESS may take on a customed appearance. This is especially helpful when an often used control character is difficult to reach.

To remap the key, an assignment statement must be made. This assignment is done as follows:

Control key letter = Function letter

I.E. to map CTRL A to go to the END instead of the TOP of the file, the following statement is used:

"A = Z"

Control key "A" gets function "Z".

Note that after a key has been remapped, the original function is no longer assigned. To remedy this, the original function may be remapped to the key of the new function i.e. from the case above:

"Z = A"

key "Z" gets function "A". Now CTRL Z goes to the TOP of the file while CTRL A goes to the END of the file.

Because of the way keys may be remapped, it is possible to duplicate functions with more than one key. This also means that it is possible to disallow functions alltogether. This ability to duplicate and disable functions allows the administrator to set up an editor tailored for special applications.

The help functions always reflect the current mapping of the keyboard.

2.6.4 TERMCAP FILE

The TERMCAP file contains the codes and capabilities that TESS needs to operate a non-HP terminal. Each type of non-HP terminal must have it's own TERMCAP file. CCSC supplies a number of these files in the group "TERMCAP.CCSCSOFT". To create your own TERMCAP file, refer to §6.7 of the TESS Reference Manual.

2.6.5 COMMAND FILE

The Command File contains multi-function strings which have been saved so that they may be recalled when needed (see §5.3.21). Each record of the file is divided into a *comment field* and a *command field* with a vertical bar ("|") separating them. The file should be unnumbered with a record length of 80 characters and a file code of 111. A sample command file is supplied in CMDFILE.DATA.CCSCSOFT.

2.7 END OF FILE MARK

The special mark, **EOF**, is used to denote the end of file while using TESS. The cursor is never allowed to go onto or beyond this mark. If the user moves the cursor onto the mark, a line will automatically be inserted and the cursor placed on the inserted line. Thus, all handling of the mark is taken care of by TESS.

2.8 TESS VERSION NUMBER

A TESS banner is displayed every time the program is initiated. Contained in the banner is a Version Letter, Release Number, Revision Date (in parenthesis), and a copyright statement. If a demo version is installed, a "D" is placed after the release number. Customers under maintenance are sent each release after they have been approved.

3.0 UDC IMPLEMENTATION

The following UDC should be used for the system UDC.

```
EDIT !EDITFILE,!FASTEXT=$NULL, !INFO="<info string>"
FILE EDITFILE=!EDITFILE
FILE FASTEXT=!FASTEXT
FILE EDITLIST;DEV=LP
RUN TESS.UTIL.SYS;INFO="!INFO"
```

NOTE: This UDC **does not** use the fast text option, so a fast text file is not left in the group (see §3.2.2 Fastext option).

The following UDC should be used for user UDC's.

```
EDIT !EDITFILE=<mywork>,!FASTEXT=<mywork>,&
!INFO="<info string>"
FILE EDITFILE=!EDITFILE
FILE FASTEXT=!FASTEXT
FILE EDITLIST;DEV=LP
FILE TERMCAP=<termcapfile.termcap.ccscsoft>
RUN TESS.UTIL.SYS;INFO="!INFO"
```

NOTE: This UDC **does** use the fast text option. <MYWORK> should be a unique file name reserved only for fast text files.

NOTE: Only use a TERMCAP file equation if UDC is intended for non-HP Terminals.

If the first line of the UDC must continue onto the next line, an ampersand is required at the end of the first line. This will effectively concatenate the first and second lines in the UDC definition.

3.1.1 INFO OPTIONS

The keywords for the INFO option are as follows:

NOKSAM	**NOTRUNC**	**UNN**	**TIMINGS**
NOPRIV	**NOWARN**	**NOBREAK**	**NEWUNN**
HIPRI	**VHIPRI**	**TERM18**	**TAE**
CHARS=<nnn>	**AVAIL**=<nnnnn>		

NOKSAM will not allow the texting of KSAM files. Normally, they may be opened with a warning message. If kept, the keys will be lost.

NOTRUNC will not allow records greater than the screen size to be edited. Normally, files up to 256 characters may be opened with a warning message. If kept, the records will be truncated.

UNN will assume the file to be un-numbered. Normally, a file with 8 numbers on the rear or 6 numbers on the beginning of the first record will be opened as a numbered file and these numbers will be suppressed. New files will be unnumbered as well.

NEWUNN is valid only on new files. When a file is new and NEWUNN is in effect, the file will be un-numbered. The default is the file is numbered.

TIMINGS turns on the TESS end of job timings for CPU and connect seconds.

NOPRIV turns off NOWAIT IO to the Terminal. Response time in character mode will be slightly slower, since the privileged opening of the Terminal will be bypassed and normal WAITED IO will be used.

NOWARN turns off the minor warning messages and only a bell sounds.

NOBREAK turns off the ability to break out of TESS via the system break key. This is helpful if the break key is often hit accidentally.

HIPRI and VHIPRI boost the program priority in the system queue to linear 'C' and linear 'B' respectively only when characters are being typed. This should only be used if a heavily loaded system degrades character key response.

TERM18 sets up the Terminal for faster communication (leaving off some redundant handshaking codes). This option should only be used with Terminals which are "fast" enough to handle it (e.g. 26XX).

TAE notifies TESS that a TYPE AHEAD ENGINE (mfg. by the Type Ahead Engine Company) is online between the computer and the Terminal.

CHARS is the character length desired on a new file. If not specified, then 72 will be used. This figure is always in bytes and does not include any line numbers.

AVAIL is the amount of available records for addition to the work area. This parameter is used when joining a large file into the work area. Normally the following algorithm applies:

```
If EDITFILE EOF is less than 1000 then LIMIT = 2000
ELSE
    If EDITFILE EOF is less than 4000 then LIMIT = 1.5(EDITFILE EOF)
    ELSE
        LIMIT = EDITFILE EOF + 2000
AVAIL = LIMIT - EDITFILE EOF (LIMIT must be less than 32,768)
```

The keywords may appear in any order, in upper or lower case, and if duplicated, the last duplicate keyword will be used.

Examples:

EDIT NEWFILE,,"timings"
Will create a 72 character numbered file named NEWFILE along with a default Workfile if so specified in the UDC and give timings.

EDIT NEWFILE,WORK,"CHARS = 80,UNN"
Will create an 80 byte un-numbered file named NEWFILE, and a Workfile named work.

EDIT OLDFILE,,"NOKSAM,NOTRUNC"
Will text up the file named OLDFILE as long as it is not KSAM, and the record length is not wider than the Terminal screen.

3.1.2 FASTEXT OPTION

TESS allows a unique system of re-editing a file. Once a file has been edited in TESS, a FASTEXT file may be left behind, provided it is not sent to $NULL. This FASTEXT file is the last 'workarea' of the EDITFILE. Each time a file is edited, a FASTEXT file is created. It assumes an 'E0000000' file name until keep time when a rename is done to either a UDC specified FASTEXT name or to $NULL. Since the workarea already exists on successive edits, there is no time spent texting-up the file.

The option to have a faster text may be exercised by including a file equation which equates FASTEXT to a user specified name that will be used exclusively for re-editing the last edited file.

For example EDIT TEXT,FAST will edit a file called TEXT and the work area will be stored under the name FAST. Next time, to quickly re-edit TEXT the user need only type EDIT FAST. At keep time, the changes will be posted to TEXT, and FAST will also be kept.

If there is only one EDITFILE to be considered in the group, only one FASTEXT file is needed, and the user UDC may be incorporated. This UDC designates the same default name for the FASTEXT file as for the EDITFILE. Therefore, it may be assumed when EDIT is typed in without any specific filename, that the user wants to open the last workarea eg. the default FASTEXT file. At keep time, the changes are made in the EDITFILE and the workarea will again be given the default FASTEXT name. In another words, the first time the file is edited by typing EDIT TEXT. Subsequent edits may be quickly accomplished by typing EDIT (see §4.3 Re-Texting). If there is more than one EDITFILE to be edited within the group, this technique can still be used, but it must be kept in mind that the default FASTEXT file will always be the last edited file.

NOTE: After a line has been inserted or deleted, there will be a purge and rebuild at keep time. Once this has occurred in the workarea, it will continue to do a purge and rebuild on successive keeps of the workarea. Time-wise, an 'update' keep of only the modified lines is faster than a complete keep and a 'fast' text of the workarea is faster than a complete text. These time differences become more vital as the file gets larger. A trade-off exists when lines are inserted or deleted and a complete keep must be performed. The workarea will be instantly texted but not instantly kept. Since a complete text is faster than a complete keep, it is preferable to do a complete text in this situation. This will purge out the old workarea and replace it with a new one and, assuming that there will not be any insertions or deletions in the next edit session, the overall elapsed time will improve.

4.0 EDITING PROCESS

4.1 STARTUP—TEXTING

The command to edit a file is:

```
EDIT <filename>,<workname>,<info string>
```

Where <filename> is the file to be edited, <workname> is the name of the work area to be saved, if any, and <info string> contains the file texting information ie. character length of a new file. If the INFO string is not included, the default is used as set up in the active UDC (see §3.1.1 Info Options).

TESS attempts to gain exclusive access to the edit file. If the file cannot be locked, it is opened shared. At keep time, in this situation, the user is asked to rename the modified file to a unique name.

Upon initiating the edit sequence after the program is loaded, a short header follows stating what version and what time TESS is running. After the file has been texted, information about the file and the current TESS environment is supplied.

For Example:

```
:EDIT TEXT,,"CHARS=80"

TESS [A.7.0] MON, NOV 21, 1983, 5:54 PM
Copyright CCSC, Inc. 1981

Files:
    Work: E2117592
    Edit: TEXT.PUB.SYS
Domain and Access: New file, Shared Read access.
Line Number Style: Numbered.
Date/Time Created: 11/21/83 - 17:54
Size (Characters): 80
Number of Records: 1
Available Records: 1999

Press RETURN:
```

There are five different ways to start the editing process:

<CR>	Top of file.
^A	Top of file.
^Z	Bottom of file.
^T	Go to an absolute record number.
^Q	Quit the program.

NOTE: If a file is new, there will be one line and the EOF marker. TESS will not allow a file to become empty.

Editing functions may now be performed on the file.

4.2 ENDING A TESS SESSION

The 'work file' keeps track of the name of its 'edit file' (ie: EDITFILE). At keep time, TESS either updates or purges the EDITFILE and rebuilds it with the contents of the work file.

4.2.1 KEEPING

When exiting the program (see §5.2.18 Quit and/or Keep) a message is given asking the user if the old file should be purged. Upon an affirmative response, the file is kept with the new changes, and the program ends. If the file is new, the statement: Saving <edit file> is given indicating that the file is being kept for the first time.

Examples:

```
                    <Old file>

Quit TESS:

Replace old file MANUAL.PUB.SYS? y

Lines Kept =

END OF PROGRAM
:

                    <New file>

Quit TESS:

Save new MANUAL.PUB.SYS? y

Lines Kept =

END OF PROGRAM
:
```

4.2.2 QUITTING

If quitting the program was not intended, or the changes are not wanted, a quit option is available.

Example:

```
Quit TESS:

Purge old MANUAL.PUB.SYS? n (or <CR>)

Quit without posting changes? y

END OF PROGRAM
:
```

If no changes were made then the program will only prompt for quitting.

4.2.3 UPDATE KEEP

If no lines have been inserted or deleted then the file is merely updated with the modified lines. This is very helpful when only simple modifications are being made to a file.

Example:

```
Quit TESS:

Update old file MANUAL.PUB.SYS? y (or <CR>)

Lines Modified =

END OF PROGRAM
:
```

4.2.4 RENAMING

Often it is necessary to rename the Workfile's keep name so that the EDITFILE will be left intact. This is accomplished by using the rename key (see §5.3.18 Rename the Keep File Name) and by giving the file a new name. The EDITFILE is essentially being reassigned, so if the new name is an old file, TESS will attempt to open it exclusively just as before.

Example:

```
Rename Editfile:

EDITFILE = MANUAL.PUB.SYS

NEW FILENAME = <New file>
```

NOTE: A carriage return will leave the file name unchanged.

4.3 RE-TEXTING

To re-text a file which has been texted at least once within the current group, use:

```
:EDIT <CR>
```

This command will open the old Workfile (formal designator FASTEXT) which was left in the group upon TESS's last completion. Texting this Workfile results in a fast text. Since the workarea is already established. This option becomes more effective with larger files.

NOTE: Only the user UDC (see §3.1 UDC Implementation) enables this option. Use of the system UDC will not leave a fast text file in the group and thus will not allow for fast re-texting.

4.4 RECOVERY

Should an interruption occur during editing, recovery of the work area is always possible with the exception of the most recent changes on the screen. This recovery is attempted by editing the 'E File' which is left in the group (see §2.6.1 Workfile).

4.5 TEXTING WITHIN TESS

Another process of TESS may be launched through the MPE key. (See §5.2.31). By typing CTRL ^, causing a programmatic break to MPE, and then typing the normal "EDIT <filename<", TESS will suspend itself and launch another TESS process. Once editing of the

second file is complete, and CTRL Q has been issued, the previous process is reactivated. A carriage return and CTRL S will restore the screen to that of the first file.

5.0 COMMAND SET

There are 64 possible commands available to the user via the CTRL key. A command is made by pressing CTRL simultaneously with the appropriate key. (The shift lock is inconsequential since CTRL characters are unaffected by case.) Each control character is capable of generating one of two possible commands: codes preceded by a CTRL \ are interpreted as Tier 2 keys and non prefixed codes are Tier 1 keys. The most common functions are found on Tier 1. Not all keys are mnemonic representations. The accent is on key placement rather than aesthetics. Most of the commands are grouped by families.

NOTE: On all commands, the bell sound denotes an error or warning. A message then follows with the prompt 'Press Return:'. At this point the program could not execute the chosen command and nothing was changed. Any response acknowledges the message and restores the screen.

5.1 RESETTING COMMAND PARAMETERS

^X is used to activate the parameter dialogue for the next command. Some commands do not have parameters and they are unaffected by this key. Once the control X is used, the next function that requires parameters, will prompt the user for new constants. The input is explained under each key Section.

5.2 TIER I (Regular Command Set)

5.2.1 ^\ SECOND TIER FUNCTION SHIFT

This key places TESS into the second set of commands much like a character shift key. No acknowledgement is given, until after it has been suffixed by another control key.

CONTROL X: Not an option.

5.2.2 ^A DISPLAY BEGINNING OF FILE

This key displays the first 24 lines of the file. If the file contains less than 24 lines then the file will be shown with the end of file mark after the last line in the file.

CONTROL X: Not an option.

5.2.3 ^B MOVE DOWN 12 LINES

This is a fast method of moving the cursor down 12 lines. If the cursor hits the bottom of the screen, then the screen will scroll up rolling text off of the top and gaining text at the bottom until the EOF.

CONTROL X: Not an option.

5.2.4 ^C **CURSOR UPDATE**

When the system is under a heavy I/O load, cursor movement is often slowed down. For faster cursor movement, the local cursor keys may be used. Once the cursor is positioned, a ^C must be issued in order for TESS to know where the cursor has moved.

WARNING: FAILURE TO USE CTRL C AFTER LOCALLY MOVING THE CURSOR WILL CAUSE UNPREDICTABLE RESULTS.

CONTROL X: Not an option.

5.2.5 ^D **DELETE CHARACTER**

To delete a character, position the cursor under the character and press ^D. The character will disappear and the line will automatically shift over to take up the excess space.

CONTROL X: Not an option.

5.2.6 ^E **INSERT CHARACTER MODE**

Unlike most other commands, this one places the Terminal into a mode until another command is issued. This instruction also acts like a switch: if hit once it turns insert characters on, if hit again, it turns insert characters off. It is not necessary to explicitly turn off insert character mode with the ^E command. All other commands turn insert mode off before execution.

NOTE: Most terminals are equipped with some means of indicating that the insert character mode is on. It is often helpful to watch that indicator.

CONTROL X: Not an option.

5.2.7 ^F **FORWARD FIND AND TAB**

The 'find' command searches through the file from the point of the cursor until it finds the matching string. If no match is found or the find is stopped, a warning is given, and the cursor stays where it was before the find was initiated. If a match is found on the screen, the cursor is moved to it. If a match is found which is not on the screen, the matched line is positioned at the top of the screen.

There are four conditions which will stop a forward find: (1) A ^Y is sent by the user, (2) the *target string* is not found before the end block mark, (3) the target string is not found before the end of file mark, or (4) the target string is found.

Forward and reverse 'finds', will find either (1) an explicit character string (with or without case constraints), or (2) a character string which matches a pattern (or one of a series of alternative patterns). The string or pattern is entered in response to the 'Target String' prompt. To search for occurrences of an explicit character string, enter the string, undelimited.

To search for occurrences of a string regardless of case (a *caseless* find), prefix the string with a tilda ("~"). This means that a Target String of "~ABCDEF" will match a string like "aBcdEf".

To search for occurrences of strings which match a pattern, or one of a series of alternative patterns, begin with a vertical bar and separate each alternative pattern with vertical bars. Each pattern may combine literals (strings enclosed by double quotation marks), 'wild card' pattern characters (listed below, may be preceded by a definite or indefinite repetition factor), or the decimal equivalent for a specific ASCII character (numerics enclosed by angle brackets).

Available 'wild card' characters include:

A [upper or lower case alpha]	N [numeric]
U [upper case alpha]	P [punctuation, including space]
L [lower case alpha]	V [any printable ASCII character]

Each wild card may be preceded by a one or two-digit repetition factor or by a period, which indicates indefinite repetition. (Examples: the pattern |"F".VP would find 'FOPEN(', 'FREAD,' 'FSTART:' etc., while the pattern series |"IF" |"THEN" |"ELSE" would find all occurrences of just these three words).

Additionally, the cursor may be tabbed over a certain amount of columns to the right of the first letter of the match, and the search may be restricted to a column range within each line. The defaults for these two values are zero tabbing and the entire string will be searched. These values are input as follows and they stay constant until reset with the ^X function.

An option to the "Range" prompt is the ability to find text which occurs only at the begin or end of a line. To invoke these options, enter a "BEGIN" ("B") or "END" ("E") in response to the "Column Range:" prompt. If "B" is entered, TESS will search the deblanked beginning of each record for the Target String. IF "E" is entered, TESS will search the deblanked end of each record for the Target String (matching only if the last character of the string is the last character of the line.)

Target String:

\<string>, <~string>, <|>, or <CR>

Tab:

\<numeric> or <CR>

Column Range:

\<First/Last>, <B[EGIN]>, <E[ND]> or <CR>

CONTROL X: Allows resetting of the string, tab, and search range.

5.2.8 ^G **GRAB BLOCK**

The 'grab' function is used to move or delete blocks of text. A block of text is defined in §5.2.28 and §5.2.30. If a block is not defined then a warning will be given, otherwise the block of text will be extracted from the file and held. Once the block is held, the cursor may be moved elsewhere in the file and the held block may be inserted (see §5.2.23 Insert Grabbed Block).

At keep time, TESS will ask the user if the grabbed block may be deleted. A negative response will allow the user to re-enter the file and insert the block wherever desired. However, if a positive response is given, the block is permanently deleted.

Grabbing the whole file will result in a hold of all the lines of text and a creation of one lone line since TESS does not permit an empty file.

CONTROL X: Not an option.

5.2.9 ^H **CURSOR LEFT**

This key moves the cursor left one position. If the cursor is on the beginning of the line, the cursor will be positioned at the logical end of the previous line. If the previous line is the top of the file, then the cursor is positioned at the end of the present line.

Execution of this command may be accomplished by the backspace key as well.

CONTROL X: Not an option.

5.2.10 ^I **TAB HORIZONTAL**

There are six different types of tabbing available: 'Word', 'Alpha', 'Numeric', 'Special', Absolute, or Relative. Tabbing to the next 'Word' is the default setting unless the Terminal's absolute tab stops are set prior to TESS initialization. If Terminal tab stops are set, these positions become the default settings at start up time. Of course, the tab definition may be changed at any time to tab to the next 'Word', 'Alpha', 'Numeric', 'Special' character or any combination of these or to an absolute or relative column number.

When entering the tab information, only the first letter needs to be used and commas are not required except for absolute and relative tabbing (e.g. "AN", or "NS", or "10,25,50"). In these cases, the relative numbers are differentiated from the absolute ones by their sign, either a plus or a minus (e.g. "10,20,-5,30,+3"). Absolute (or 'Typewriter' type) tabs are automatically set in both CHARACTER and ADD modes. There are some limitations to remember in ADD mode, however. (1) Do not backtab. (2) Do not tab past the last tab set. If either of these situations occur, exit from ADD mode and refresh the screen.

To set tabs, use the ^X and input as follows:

```
SET TABS:

\<a><n><s><w> or <10>,<20>, or <+10>,<-20> or <CR>
```

Execution of this command may be accomplished by the tab key as well.

CONTROL X: Allows the tab key to be reset.

5.2.11 ^J **CURSOR RIGHT**

This command moves the cursor to the right one position. If the cursor is on the physical end of the line then the command will reposition the cursor to the first column of the next line. When the next line happens to be the end of the file, a new line opens up and the cursor will move to the first column of that line.

CONTROL X: Not an option.

5.2.12 ^K **KILL LINE FROM CURSOR**

When evoked, all characters to the right of and including the cursor character will be deleted from the screen.

CONTROL X: Not an option.

5.2.13 ^L **DELETE LINE**

This key deletes the line the cursor is on, bringing up a line of text or the EOF mark from the bottom of the screen. If the line to be deleted is the only line in the file, then only a kill line is performed. The last 24 deleted lines are kept on a last-in first-out stack in memory in case a recently deleted line must be brought back (see §5.3.22 Undo).

CONTROL X: Not an option.

5.2.14 ^M CURSOR RETURN

The RETURN key and the ^M are one and the same which makes the ^M rarely used. Depressing the return key, or the ^M will cause the cursor to return to the first column and line feed. When the cursor is at the bottom of the screen then the text will roll up one line. If, however, the cursor is above the EOF nothing will happen. The cursor will not move since it may not move onto the EOF. The RETURN key may be re-defined to any other function by using the second tier ^M.

CONTROL X: Not an option.

5.2.15 ^N CURSOR DOWN

Pressing the ^N will move the cursor down to the next line while remaining in the same column. If the cursor is at the bottom of the screen, the text will roll up one line exposing the next line of text or the EOF. When the EOF rolls up, the cursor will remain on the bottom line. If ^N is executed above the EOF mark on the screen, there will be a new line inserted and the cursor will be positioned in the same column as it was in the previous line.

CONTROL X: Not an option.

5.2.16 ^O INSERT LINE

This key will insert a line above the current line. Lines below the inserted line will roll down one. The cursor will tab over automatically to the margin of the current line. When more than one line needs to be inserted, ADD mode should be used (see §5.2.32 Add Mode).

CONTROL X: Not an option.

5.2.17 ^P STATUS

From time to time the user may want to know some particular status information about the TESS environment. The same information is given as in the text up status (see §4.1 Start-up—Texting) plus information about: TESS deactivation (if a demo version), current tab settings, the current find string, and the present position in the file and on the screen.

NOTE: The file position (referred to as Line: <line number>), is the absolute line number at text time. Insertions and deletions are not reflected in this number.

The user can also display the contents of the two *hold areas* available in TESS. The first area shown is an abbreviation of the *grab buffer* (see §5.2.8) and the second area is a full list of the last 24 deleted lines.

CONTROL X: Not an option.

5.2.18 ^Q QUIT and/or KEEP

This key is used to quit editing and, if desired, keep the file (see §4.2 Ending a TESS Session).

CONTROL X: Not an option.

5.2.19 ^R **REVERSE FIND and TAB**

The reverse find accomplishes the same thing as the forward find only it searches backwards through the file. The same search string is used. (see §5.2.7 Forward Find and Tab.)

CONTROL X: Is used to reset the search string and tab.

5.2.20 ^S **SHOW MEMORY**

If for any reason the screen is locally lost or corrupted, this command will refresh the screen. Each line is printed to the Terminal and the cursor is positioned to where TESS believes it to be. Notice that when a full screen is written, it is done with a single write to the Terminal and leading blanks are skipped.

NOTE: TESS does not turn messages off. This is left up to the user to put into the UDC if it is so desired.

CONTROL X: Not an option.

5.2.21 ^T **TRAVEL**

The Travel key brings up a screen of the file from a requested absolute or relative record number. If a relative travel is necessary, the input should have a plus or minus sign before the number of relative records to move. Without the sign, the input is assumed to be absolute movement, relative to the first record of the file, record one. ^Y will break the travel and leave the cursor where it was before the command.

To Travel directly to an absolute line number in the file (as it was at text time) insert a "@" before the line number. This will make the Travel happen immediately. This feature is especially useful when travelling to a particular line after additions or deletions have been made in the file causing relative line numbers to change. NOTE: Care must be taken not to travel to a Grabbed or deleted line. Presently, the deleted line will appear on the screen but the Keep and any other access will ignore this line.

Additionally, a travel may be undone by typing 'UNDO' after the travel prompt. This will restore the screen to where the user last traveled from.
The input is as follows:

Record Display:

\<1> or <+1> or <CR>

CONTROL X: Not an option.

5.2.22 ^U **CURSOR UP**

With this command, the cursor may be moved up one line. If the cursor is at the top of the screen, then an attempt is made to roll the text down one line. However, if the top of the file has been reached, nothing will happen. The cursor always remains in the same column.

CONTROL X: Not an option.

5.2.23 ^V **INSERT GRABBED BLOCK**

After a block has been grabbed (see §5.2.8 Grab Block) the user may go anywhere in the file and insert that block below the current line, denoted by the cursor. Once the block has been inserted, the held block is empty, and other blocks may be grabbed. If the held block needs to be inserted above the first line of the file, a dummy line must first be inserted (see §5.2.16 Insert Line) and then the INSERT may occur below the dummy line. After the INSERT is complete the dummy line may be deleted (see §5.2.13 Delete Line).

CONTROL X: Not an option.

5.2.24 ^W **WRITE TO EDITLIST**

The user may send a block or all of the file to the list file called EDITLIST with the ^W. If the file is numbered, numbers will appear on the listing corresponding to the absolute record numbers at the time of the writing. A banner will also appear at the top of each page telling the page number, the name of the file, and the date. When a $PAGE or !PAGE is found flush left on a line, a page break is generated and the $PAGE suppressed. A user defined banner will replace the default banner if a quoted string is found after the $PAGE command (e.g. $PAGE "My Heading").

CONTROL X: Supresses all banners (including page numbers).

5.2.25 ^X **RESET PARAMETERS FLAG**

The ^X is used prior to a variety of commands to allow resetting of information generic to the key. When the reset flag is on and a control key is executed, the user will be prompted for new parameters.

5.2.26 ^Y **MOVE UP 12 LINES**

This command will move the cursor up 12 lines on the screen. If the cursor reaches the top, the text will roll down until the top of the file is reached. It is a faster cursor movement than can be achieved by repeating a ^ U.

5.2.27 ^Z **BOTTOM OF FILE**

This command will take the user to the bottom of the file, displaying only the last line and the EOF.

CONTROL X: Not an option.

5.2.28 ^[**BEGIN BLOCK**

The begin block key may be issued regardless of where the cursor is resting in a line. It will erase any previously set beginning block mark and mark the first character in the current line with an inverse video box. This defines the current record as the beginning of a block of data for gathering, copying, etc. Only one block may exist at a time.

CONTROL X: Turns off the begin block if it is turned on in the file.

5.2.29 ^@ **HELP TIER I**

The HELP key will list the first tier commands in ASCII order. After the list is complete, the user may type any character for further information.

CONTROL X: Not an option.

5.2.30 ^[**END BLOCK**

The end block acts the same as the begin block key (see §5.2.28 Begin Block) except that it marks the last character in the current line with an inverse video box. This defines the current record as the end of a block of data for gathering, copying, etc. Only one block may exist at a time

CONTROL X: Turns off the end block if it is turned on in the file.

5.2.31 ^^ **MPE COMMAND INTERPRETER**

This key allows you to execute MPE commands including the PREP and RUN commands. For example:

```
MPE Command:

:PREP $OLDPASS,$NEWPASS or just :PREP

:RUN LISTEQ2.PUB.SYS

:FILE INPUT=INDATA

:RUN $OLDPASS or just :RUN

:EDIT INPUT

:DEBUG

:FCOPY

:<CR>
```

The :PREP command by itself defaults to :PREP $OLDPASS,$NEWPASS as does the :RUN command. Also, for installations with VESOFT'S MPEX facility, any MPEX command may be executed from within TESS using this key by preceding the MPEX command with a percent sign ("%").

CONTROL X: Not an option.

5.2.32 ^_ ADD MODE

To add more than one new line of input, the add mode is preferred. This mode is a line mode, not the normal control character mode. Within this mode, control characters are invalid and are thrown out with the exception of ^Y, TAB, BACKSPACE and RETURN. Tabs are exploded to blanks.

In ADD mode, characters are input until a carriage return or the end of the input record is reached. At this point a carriage return line feed is generated, and one line will open up below the current line. This new line is marked in inverse video.

To return to character mode, press carriage return and ^Y.

CONTROL X: Not an option.

5.3 TIER II (extended command set)

5.3.1 ^\ CANCEL SECOND TIER

All second tier commands cancel the second tier and place the user back into the first tier automatically. If the second tier prefix is inadvertently pressed, the user may press it again to cancel the second tier prefix.

CONTROL X: Not an option.

5.3.2 ^A RAISE CURRENT LINE

The current line that the cursor is resting on, may be raised to the top of the screen and thereby bring up a screen of text from that point.

CONTROL X: Not an option.

5.3.3 ^B NEXT PAGE

Paging through the file is accomplished with this command. It will bring up the next 24 lines following the present screen of text. If no more lines can be brought up due to the EOF, then a warning will be given. The cursor is repositioned at the top of the screen.

CONTROL X: Not an option.

5.3.4 ^C COPY BLOCK

To copy text, the user must first mark the block to be copied (see §5.2.28 Begin Block) and then move the cursor to the line above the destination position. The copy will insert the block below the current line. To copy a block to the top of a file, a dummy line must first be inserted as in the ^V command (see §5.2.23 Insert Grabbed Block). ^Y will halt the copy, but some text will have been added to the file.

CONTROL X: Not an option.

5.3.5 ^D SHIFT BLOCK LEFT

This command deletes indention for a block of text. Text cannot be lost. If the amount of spaces to shift exceeds the amount of spaces available at the beginning of the line, it will only shift as far as possible and the line will be flush left. The shift count remains in effect until reset. If no blocks are set then only the current line will be shifted. TESS prompts the user for the shift count the first time as follows:

SHIFT COUNT:

\<2> or <CR>

CONTROL X: Will allow resetting of the shift count.

5.3.6 ^E SHIFT BLOCK RIGHT

This command allows indenting of a block of text. Both Begin and End block marks must be set. Text cannot be lost. If the amount of spaces to indent exceeds the amount of spaces available at the end of the line, it will only indent as far as possible and the line will be flush right. The shift count remains in effect until reset. TESS prompts the user as in the left shift.

CONTROL X: Will allow resetting of the shift count.

5.3.7 ^F SORT BLOCK

This command sorts a block of text using the system sort utility. Up to sixteen different keys may be defined as follows:

Key \ <column>,<length>,<Ascending/Descending>

The keys are one based and should be defined in order of importance. (i.e. The first definition is the primary key, the second is the first secondary key, etc.). The Ascending/Descending phrase is optional and if not included, defaults to ascending ordering. Once the sort is complete, changes are posted to the *Workfile*, and the newly sorted lines are displayed.

An alternative method for defining a key is provided through the use of the cursor. Before the sort is initiated, place the cursor at the first position of the key field. When prompted for the Key, enter a "C" (for cursor) instead of the column position. When the sort is performed, the cursor position and given length will be used as the key field.

CONTROL X: Not an option.

5.3.8 ^G USER LIBRARY ROUTINE

This command is used to load, subsequently execute or unload a library routine. For loading or unloading, TESS will prompt as follows:

User Library Routine:

\<routine name> or <CR>

At this point, TESS searches the group and then the account and finally the system SL for the requested routine, and links it into the presently running program. Subsequent use of the key

will then execute the routine. Once the program terminates or upon user request, the routine is released. For further information on writing a TESS routine, see the appendix.

CONTROL X: Releases the routine.

5.3.9 ^H PROGRAM COMPILATION

This key allows program compilation without keeping. After the key is pressed, the user is prompted for a compiler name. When SPL, FORTRAN, RPG, TRANSACT, PASCAL or COBOL[1] is typed, the text will be compiled, including any previous changes, with the listing sent to $NULL. CTRL Y will cleanly abort the compilation and return to the file. To redirect the listing, a file equation must be issued. For example, if the listing is to go to the line printer, the file statement would read:

```
:FILE SPLLIST;DEV=LP
```

(see §5.2.31-- MPE Command Interpreter). The other list file designators are COBLIST, FTNLIST, RPGLIST, PASLIST and TRANLIST. The EDITFILE file remains untouched throughout the process; the compile actually works on a temporary Message file. A keep is not necessary.

Next the user is prompted for the queue in which compilation will take place. The choices are CS or DS queues. The CS queue has a higher priority and so will run faster while competing for system resources. The DS queue, on the other hand, runs as a background process using system resources only when available. TESS compilation defaults to the CS queue.

[1]There may be more than one COBOL compiler on the system and so the user may need to type the precise name if other than COBOL, ie. COBOL1, COBOL2, COBOLII, COBOLXYZ, etc.

An example of the practical use of this key might be similar to the following:

(1) :EDIT a fast text 10000 line source file (happens immediately).
(2) An adjustment in the source code is made.
(3) The Compile key is invoked.
(4) If the compile is not clean go to (2).
(5) The MPE key is invoked.
(6) :PREP calls the segmenter to prepare the USL.
(7) :RUN calls the loader to run the object code.
(8) If the run is not correct go to (2).
(9) :PURGE the old program.
(10) :SAVE the new program.
(11) Invoke the Quit command and the old source is replaced.

Note that every time it is necessary to recompile, both a Text and a Keep are avoided.

CONTROL X: Allows resetting of the compiler name and queue.

5.3.10 ^I **REVERSE TAB**

This is the back-tab function. It will tab in the opposite direction of the first tier ^I (forward tab). The parameters are set in the first tier tab.

CONTROL X: Not an option.

5.3.11 ^J **JOIN**

This key joins a file below the current line. After positioning the cursor between the two lines where the joined text is to be placed, the ^J is pressed. TESS will then prompt the user for a file name and a range as follows:

```
Join File:

\<file name> or <CR>
Join file record range FIRST/LAST, 'ALL' or RTN \
                                                  <ALL>
                                                  <1/2>
                                                  <1.1/2.4>
                                                  or <CR>
```

The numerical FIRST/LAST range may be either an absolute record number range or an editor line number range. Editor ranges must include a decimal point while absolute record ranges must not. The number of joined lines must be less than the number of available lines listed by the physical status command. A count of the lines joined is maintained every ten lines. Control Y can stop the joining; however, some lines will be joined and left in the file.

CONTROL X: Not an option.

5.3.12 ^K **ERASE TO CURSOR**

Everything up to the cursor is blanked out and the cursor remains in position.

CONTROL X: Not an option.

5.3.13 ^L **LINK LINES**

This key moves the line below the current line up to the current line, in total, in part, or not at all, depending on whether the text on the next line will fit. If the next line is not on the screen or if it is the EOF then a warning will be given, indicating that nothing was linked. The command will attempt to break the next line at the best place, either a blank or some punctuation.

Linking will add a single blank to the last non-blank character of the current line and will de-blank the front of the next line. Any margin on the next line is preserved so that when text is removed in part, the rest of the line will begin at the old margin.

CONTROL X: Not an option.

5.3.14 ^M RE-DEFINE RETURN KEY

TESS allows the return key to be re-defined much the same way that the Terminal does, but TESS does not re-define the hardware return. Only the software interpretation of the return key is re-defined. This is particularly useful when a command needs to be operated frequently and the carriage return function of line feed and carriage return is not needed. If the last function hit is needed to be repeated many times, for instance the paging key, REPEAT may be specified. This will cause the RETURN key to repeat the last used key. The prompt for setting the key is as follows:

```
Re-define Return Key:

\<character> or <\character> <CR> or 'REP'
```

Where the character represents the function key, the \ sign denotes a second tier function and CR is to leave the key a carriage return.

CONTROL X: Allows the user to reset the return key.

5.3.15 ^N LINE NUMBERING

There are three separate and distinct numbering styles available: COBOL, NUMBERED, and UNNUMBERED. COBOL style numbering reserves six digits at the front of the line. NUMBERED style numbering reserves eight digits at the end of the line. UNNUMBERED style numbering reserves no positions for numbering purposes. In any case, line numbers are not assigned until keep time. The prompt for setting the desired numbering style is as follows:

```
Number Format:
NUMBERED, UNNUMBERED, or COBOL format
\
```

Simply enter the first character of the style you require and press the return key (e.g. "C" for COBOL numbering style).

CONTROL X: Not an option.

5.3.16 ^O SPLIT LINE

When new text needs to be added to a line already full, the split line command is used. By positioning the cursor on the character destined for the new line and pressing ^\^O, the text will be split at that point. A new line is actually inserted below the current line and the second part of the current line is moved into it. This is all done maintaining the margin of spaces. The cursor comes to rest at the point of the split and typing may resume.

CONTROL X: Not an option.

5.3.17 ^P APPEND TO A PENDFILE

A block of text may be appended to a file of the users choosing by executing the append key. Initially the pendfile name must be provided as follows:

```
Pendfile Name= <pendfile name> or <CR>
```

This will open a new or old file for appending with the appropriate name. After the block has been set, ^\^P may be pressed again and the block of text will be copied to the pendfile. The file may hold up to 32,768 records.

CONTROL X: Closes the pendfile and allows opening of a new one.

5.3.18 ^Q RENAME THE KEEP FILE NAME

After texting up the file into the work area, it may be desirable to create a new file to keep the text in, so as not to replace the original data. The rename key accomplishes this by prompting as follows:

Old Editfile = OLD EDITFILE NAME

New Editfile = <new editfile name> or <CR>

Warnings are issued when the new filename already exists or is shared. If the new file already exists, it will be replaced at keep time. If it is shared, then another name will be required.

CONTROL X: Not an option.

5.3.19 ^R REPLACE

The 'replace' key replaces a target string with a replacement string within an optional column range, inside of a text block. To replace all occurrences in the file, a begin block must be set on the first record and an end block set on the last record. The replace will not execute unless a block has been initially set.

The optional column range defaults to the first through the last column of each record. If the replacement would expand the line beyond the record length, a warning will be given and the cursor will be deposited at the beginning of the unreplaced target string. Changes from the begin block up to but not including the cursor will have been successful.

The replace key and the find keys DO NOT share the same stored target string. Control Y will halt the replace with only changes up until the control Y being made. The following is the format of the replace information:

Target String:

\<string> or <CR>

Replace String:

\<string> or <CR>

Column Range:

\<first column/last column> or <CR>

CONTROL X: Allows resetting of the replace parameters.

5.3.20 ^S **SWAP LINES**

The 'swap' key will swap the next line and the current line. If the next line is not on the screen or if it is the EOF then a warning will be given and no swap will take place.

CONTROL X: Not an option.

5.3.21 ^T **MULTI-FUNCTION**

With the Multi-function Key, you have the ability to combine multiple editing commands into one key thus allowing the automation of repetitive tasks. The first use of the key defines the command string, while subsequent uses execute the string in a left to right order.

Any of the 64 TESS functions (except \T) may be included in the string by using the corresponding letter. "Literal strings" may be placed in the text by surrounding characters with double or single quotes. (These "literals" behave as normally typed characters when executed). In addition, functions or literals may be combined into "repeat groups". Items in these groups must be surrounded by parentheses and prefixed with a repeat count or a repeat function. (See Appendix 6.6.2 for examples).

TESS currently provides seven "Special Functions" for added control over execution of the command string. These functions must be surrounded by angle brackets and are defined as follows:

⟨R⟩ — Repeat the following group until ^Y or a warning.
⟨P⟩ — Pause one second before continuing execution.
⟨[⟩ — Exit the repeat group if the cursor is on the *begin block* line.
⟨]⟩ — Exit the repeat group if the cursor is on the *end block* line.
⟨[[⟩ — If the *end block* is not set, issue a warning.
⟨]]⟩ — If the *begin block* is not set, issue a warning.
⟨[]⟩ — If *both blocks* are not set, issue a warning.

Note: The brackets are required to denote the special function.

After entering \T, there are two methods for defining command strings. One is simply to create the string ad hoc in response to the Command String prompt, and the other is to select the string from a previously created *Command File*.

If the string is created ad hoc, you will be prompted to save it in a Command File as follows:

```
Command String:

\<string> or <CR>

Add to Command File? (Y/N) Y      ("N" and <CR> do not add)
Command File Name: <filename>
Comment Field: <comment>
```

If the desired command string already exists in a command file, enter a dollar sign followed by the filename (or just a "$" to access CMDFILE.DATA.CCSCSOFT). The first "page" of the command file is then displayed on the screen. Enter "N" for next page, "P" for previous page, or the item number of the string to select. For example:

```
Command String:

\$CMDFILE.DATA.CCSCSOFT        ($<CR> Default file)

1: Justify Block    ( \[⟨R⟩(⟨R⟩(⟨L⟩⟨]⟩M)
2: Downshift Block ( ⟨[]⟩\[⟨R⟩(⟨VI⟨]⟩))

Next, Prev, or Item Number: 2     (Select Downshift Block Command)
```

The Multi-function Key is especially useful for inserting often used text at various places within the file. For a more comprehensive description of the syntax and use of this key, see §6.6.2 in the Appendix.

CONTROL X: Allows resetting of the command string.

5.3.22 ^U UNDO

The UNDO undoes the current line to the way it first appeared when it came onto the present screen. After a line rolls off of the screen, the changes are made upon the file and considered correct. The most common use of this key is to undo character modification.

Functions involving more than one line, such as GRAB, COPY, JOIN APPEND, etc. are not affected by the UNDO. To undo these operations the lines must be remanipulated with the DELETE LINE, INSERT BLOCK or other appropriate commands.

To UNDO a line deleted by ^L, the control X is used. Each deleted line is placed on a last-in first-out 24 deep stack. They may be brought back to anywhere in the text above the current line, one at a time.

CONTROL X: Used to turn on the last deleted line recovery.

5.3.23 ^V DOWNSHIFT WORD

The letters from the cursor to the next non-alphabetic are reduced to lower case. If the cursor character is not alphabetic, a warning will be given.

CONTROL X: Not an option.

5.3.24 ^W WORD SWAP

The next word is swapped with the current word from the cursor position. If following the current word there are no more words, a warning will be given.

CONTROL X: Not an option.

5.3.25 ^X CENTER LINE

This command centers the current line. If the line is already centered then a warning will be given.

CONTROL X: Not an option.

5.3.26 ^Y PREVIOUS PAGE

This command will bring up the previous page of text and reposition the cursor to the home position of the screen. If the top of the file has been reached, a warning will be given and no operation will take place.

CONTROL X: Not an option.

5.3.27 ^Z **LOWER CURRENT LINE**

The current line where the cursor resides will be lowered to the bottom of the screen causing 23 new lines of text to be rolled down. If the top of the file has been reached then a warning will be given and no operation will take place.

CONTROL X: Not an option.

5.3.28 ^] **FIND BEGIN BLOCK**

This command will locate the begin block mark and display the text from there. If the begin block is not set then a warning will be given and no operation will occur.

CONTROL X: Not an option.

5.3.29 ^@ **HELP TIER II**

A list of the extended set of commands is displayed. After the list is complete, the user may type any character for further information.

CONTROL X: Not an option.

5.3.30 ^] **FIND END BLOCK**

This command will locate the end block mark and display the text from there. If the end block is not set then a warning will be given and no operation will occur.

CONTROL X: Not an option.

5.3.31 ^^ **UPSHIFT WORD**

The characters from the cursor to the next non-alphabetic are capitalized. If the cursor character is not an alphabetic, a warning will be given and no operation will occur.

CONTROL X: Not an option.

5.3.32 ^_ **FIND END OF LINE**

It will position the cursor to the right of the last character in the line except when the line is empty or full. In these cases, the cursor is sent to rest on the last character.

CONTROL X: Not an option.

5.3.29 ^@ HELP TIER II

A list of the extended set of commands is displayed. After the list is complete, the user may type any character for further information.

CONTROL X: Not an option.

5.3.30 ^] FIND END BLOCK

This command will locate the end block mark and display the text from there. If the end block is not set then a warning will be given and no operation will occur.

CONTROL X: Not an option.

5.3.31 ^^ UPSHIFT WORD

The characters from the cursor to the next non-alphabetic are capitalized. If the cursor character is not an alphabetic, a warning will be given and no operation will occur.

CONTROL X: Not an option.

5.3.32 ^_ FIND END OF LINE

It will position the cursor to the right of the last character in the line except when the line is empty or full. In these cases, the cursor is sent to rest on the last character.

CONTROL X: Not an option.

5.4 COMMANDS REVIEWED BY FAMILY

5.4.1 MOVEMENT

Top of file	^A
Bottom of file	^Z
Current line to top	^\^A
Current line to bottom	^\^Z
Cursor:	
Up	^U
Down	^N
Right	^J
Left	^H or BACK SPACE
Carriage Return	^M or RETURN
Tab horizontally	^I or TAB
Tab reverse	^\^I or ^\ TAB
Address sensing	^C
Roll:	
Up 12 lines	^Y
Down 12 lines	^B
Next page	^\^B
Previous page	^\^Y
Travel	^T

5.4.2 SEARCHING

Forward find	^F
Reverse find	^R
Find end of line	^\^—
Find beginning block	^\^[
Find end block	^\^]

5.4.3 EDITING

Add mode	^—
End add mode	^Y
Delete char	^D
Insert char	^E
Insert line	^O
Delete line	^L
Erase:	
From cursor to end	^K
Start of line to cursor	^\^K

Link lines	^\^L
Split line	^\^O
Block markers:	
Beginning	^[
Ending	^]
Grab and hold block	^G
Insert held block	^V
Swap:	
Lines	^\^S
Words	^\^W
Sort	^\^F
Replace	^\^R
Join file	^\^J
Copy block	^\^C
Shifting:	
Block left	^\^D
Block right	^\^E
Center	^\^X
Upshift word	^\^^
Downshift word	^\^V

5.4.4 PROGRAM CONTROL

2nd Tier prefix	^\
reset	^\^\
Reset parameters flag	^X
Line numbering style	^\^N
Help	^@ OR ^\^@
Close Files:	
Keep and quit	^Q
Rename and continue	^\^Q
MPE command interpreter	^^
File status	^P
Refresh screen	^S
Return key re-defined	^\^M
Multi-Function	^\^T
Load/run library routine	^\^G
Undo	^\^U
Write block to EDITLIST	^W
Append block to PENDFILE	^\^P
Program compilation	^\^H

6.0 APPENDIX

6.1 COMMANDS BY TIER

Tier 1 Control Keys:

@	Help for first tier.
A	Display beginning of file.
B	Cursor down ½ page.
C	Cursor update (required after local movement.)
D	Delete character.
E	Insert character mode until next command.
F	Forward find and optional tab. (^X)
G	Grab and hold marked block.
H	Cursor left.
I	Tab abs, rel (+-), alpha, num, spec, word. (^X)
J	Cursor right.
K	Kill line from cursor.
L	Delete current line.
M	Carriage return line feed.
N	Cursor down.
O	Insert new line above current.
P	File status report.
Q	Quit and keep.
R	Reverse find and optional tab. (^X)
S	Show memory.
T	Travel to relative (+-) or absolute line number.
U	Cursor up.
V	Insert held block below current line.
W	Write marked block to EDITLIST. (^X)
X	Reset subsequent command's parameters.
Y	Cursor up ½ page.
Z	Display last line of the file.
[	Mark beginning of the block. (^X)
\	Second tier function prefix.
]	Mark ending of the block. (^X)
^	Escape to MPE command interpreter.
_	Add mode until CTRL Y. (^Y)

Tier 2 Control Keys:

@	Help for second tier.
A	Roll current line to top.
B	Next page.
C	Copy marked block below current line. (^X)
D	Shift marked block left. (^X)
E	Shift marked block right. (^X)
F	Sort marked block
G	Get user library routine. (^X)
H	Program compilation. (^X)
I	Reverse tab. (^X)
J	Join a file below current line.
K	Kill line to cursor.
L	Link-up current and next lines.
M	Re-define return key.
N	Line number style.
O	Open by splitting current line into two.
P	Append marked block to PENDFILE. (^X)
Q	Rename keep file and continue.
R	Replace within marked block. (^X)
S	Swap with next line.
T	Multi-Function. (^X)
U	Undo line to first state of current screen. (^X)
V	Downshift word.
W	Swap with next word.
X	Center line.
Y	Previous page.
Z	Roll current line to bottom.
[	Find beginning of the block.
\	Cancel second tier function prefix.
]	Find ending of the block.
^	Upshift word.
_	Cursor to end of line.

(^X) Reset parameters of next function.

6.2 USER KEY EXAMPLE SL PROGRAM

```
$ CONTROL USLINIT,SUBPROGRAM,MAP
BEGIN
$ CONTROL SEGMENT=ROTATE
<< ******************************** >>
   PROCEDURE  ROTATE(LINE,LEN,CURSOR,GLOB);
                                              << rotate line around cursor.
                                              >>

   BYTE ARRAY
      LINE;                                   << current line cursor is on.
                                              >>

   INTEGER
      LEN,                                    << byte line length one based.
                                              >>
      CURSOR;                                 << cursor position zero based.
                                              >>

   LOGICAL ARRAY
      GLOB;                                   << 40 words of global storage.
                                              >>

<< ******************************** >>
   BEGIN
      BYTE ARRAY
         TEMP(0:79);                          << local temporary storage.
                                              >>
      MOVE TEMP:=LINE(CURSOR),(LEN-CURSOR),2; << copy right part left >>
      MOVE  *:=LINE,(CURSOR);                 << and left part right. >>
      MOVE  LINE:=TEMP,(LEN);                 << copy back len, cursor
                                              static. >>

   END;
END;
```

6.3 TERMINALS SUPPORTED

6.3.1 HP TERMINALS

The following HP terminals are supported by TESS. Note that some terminals require the manual setting of Straps. (See the specific Terminal Reference Manual for further explanation.)

MAKE	MODEL	SOFT-KEYS	LABELS	STRAPS	ENHANCE†
HP	125	Y	N	auto	I
HP	150	Y	Y	auto	I
HP	2382	Y	Y	auto	I
HP	2621	Y	N	auto	U
HP	2622	Y	Y	auto	I
HP	2623	Y	Y	auto	I
HP	2624	Y	Y	auto	I
HP	2625	Y	Y	auto	I
HP	2626	Y	Y	auto	I
HP	2627	Y	Y	auto	I
HP	2628	Y	Y	auto	I
HP	2640	N	N	manual	I
HP	2641	Y	N	auto	I
HP	2642	Y	N	auto	I
HP	2643	Y	N	auto	I
HP	2644	N	N	manual	I
HP	2645	Y	N	auto	I
HP	2647	Y	N	auto	I
HP	2648	Y	N	auto	I
HP	2649	Y	N	auto	I
DIRECT	VP825	Y	N	auto	I
DIRECT	VP828	Y	N	auto	I

† "ENHANCE" refers to the type of enhancement used by the Terminal for begin and end block marks and add mode. "I" means inverse video is used, and "U" means underline is used.

6.3.2 Software Terminal Emulations

The following software packages run on popular workstations to emulate HP Terminals and are supported by TESS:

PC 2622-Walker Richer Quinn. (Must use F9 to remove function labels from the screen to allow the full 24 lines.)

6.3.3 Non-HP Terminals

Refer to the group TERMCAP.CCSCSOFT.

NOTE:
This list may not be exhaustive, and special changes may be made to accommodate specific HP (or other) Terminals in the future.

6.4 BUG LIST

The following is a cummulative summary of known bugs through version A.09.

DESCRIPTION	WORKAROUND
Warn 6 write exceeds file recsize when sending 132 char numbered file to LP.	Un-number the file, then print it out.
Control backspace or control tab, back tabs the cursor on screen only.	If these keys are used, use control C afterward to update the cursor location.
Control backspace or control tab in ADD mode will back tab on screen only.	No workaround. Use Control Y and Control S to refresh screen
Soft-key downloading always supresses a carriage return even when ¦ is used. Carriage return can't be achieved.	Add mode soft-key strings presently do not allow for optional carriage return. Use the softkey then the return key.
Add mode inverse video is one character less than it should be when the length is equal to either 80 or 132.	Inverse video can not be turned on for the entire line. This is a cosmetic limitation that doesn't effect the read.
Using DIRECT terminals in 132 char files will not always flip to next page entirely. 3 lines are left on the screen.	Ignore the 3 lines that may appear on the next page during dialogue. This is only a cosmetic problem.
Character response degrades when system is under heavy I/O load.	Use VHIPRI or HIPRI to place process in a higher queue during character I/O. Also use ADD mode (control _) when possible. A better solution might be to tune your system in the following areas: (1) IOQs increase to at least 128. (2) TBUFs increase to at least 15.
Port lock up when receiving a message from the computer.	Set messages off before entering TESS. If a WARN is issued, use the BREAK key to unlock.

Insert characters on a line until some fall off the right margin. Delete a few characters, CR and backspace. An extra garbage character may appear on the screen but not in the file.	Use control S to refresh the screen if this occurs.
Absolute travel (@####) allows going to deleted line.	The grab and delete keys do not mark lines as being deleted when they are removed from the WORKFILE list.
UNDO does not work on the lines above or below a just added or deleted line.	To ADD or DELETE a line requires the adjacent links to be written from memory to the WORKFILE and so the lines are not UNDOable.
If a file contains a NULL in the text (from some other source) a find will not find anything past it on the line.	Avoid inserting NULLS into the file.
Add mode loops if :EOD is input first.	Avoid typing :EOD in column 1.
ATP port lock-up when control S immediately follows another function.	ABORTIO on the device or run TERMDSM.
Compile to LP sends extraneous headers.	No workaround.

6.5 SAMPLE ENVIRONMENT FILE

```
$CONTROL SOFTKEYS
<1 NEXT     PREV  >~
<2nd tiercncl 2nd>^ \
<roll up prv page>^Y
<roll dwnnxt page>^B
< travel  tokens >^T
<indent 2in addmd>" "
<   cr   redef cr>^M
<cur addrcopy blk>^C
<2 NEXT     PREV  >~
<2nd tiercncl 2nd>^ \
<fwd find COBOL  >^F
<rev findreplace >^R
<top filetop line>^A
<bot filebot line>^Z
<left blkfind blk>^[
<rght blkfind blk>^]
<3 NEXT     PREV  >~
<2nd tiercncl 2nd>^ \
<grab blkuser lib>^G
<ins blk downshft>^V
<clr from clr to >^K
<del lineconcatnt>^L
<del charshft lft>^D
<add modefind eol>^-
<4 NEXT     PREV  >~
<2nd tiercncl 2nd>^ \
<up space  undo  >^U
<back spccompile >^H
<fwrd spc  join  >^J
<  info   append >^P
<show memswap lin>^S
<keep/end rename >^Q
$CONTROL REMAP
K=U
L=J
J=N
C=L
```

6.6 TESS SYNTAX

6.6.1 RUNNING TESS

```
<Edit command>   ::=  EDIT [<filename>] [,<workname>] [," <infostring> "]
<filename>       ::=  <file>[.<group>] [.<account>]
<workname>       ::=  {<file>[.<group>] [.<account>]} | $NULL
<infostring>     ::=  [NOKSAM] [,NOTRUNC] [,NOWARN] [,NOPRIV] [,UNN] [,TIMINGS]
                      [,AVAIL=<records>] [,CHARS=<bytes>]
<records>        ::=  1 : {32767 - filename's eof}
<bytes>          ::=  1 : {80 | 132}
```

Formal File Designators:

EDITFILE — The file to be edited. May be a *new* file or a *work* file.
FASTEXT — The *work* file. Saved for fast future access if specified.
EDITLIST — The device where text may be sent.
ENVFILE — Contains the softkey download information used at text-time.

6.6.2 MULTI-FUNCTION KEY

```
<Command Str>   ::=  <item list>
<item list>     ::=  <item> | <item><item list>
<item>          ::=  <function>  <repeat> | <cntl code> | <literal>
<function>      ::=  "⟨"<func>"⟩" [<group>]
<repeat>        ::=  <cnt><group>
<group>         ::=  "("<item list>")"
<func>          ::=  R | P | [ | ] | [[ | ]] | []
<cntl code>     ::=  {ASCII 64 thru 95}
<literal>       ::=  "<char list>" | '<char list>'
  R             ::=  {repeat group until interrupt}
  P             ::=  {pause for one second}
  [             ::=  {Exit group if cursor is at end block}
  ]             ::=  {Exit group if cursor is at begin block}
  [[            ::=  {Insure that begin block mark is set}
  ]]            ::=  {Insure that end block mark is set}
  []            ::=  {Insure that both block marks are set}
```

EXAMPLES:

1. `E'text'E`

   Insert "text" at the current cursor position.

2. `XI20(I'text'M)`

   Reset the tab stop to make sure the correct tab is entered. Place "text" at that tab position for the next 20 lines.

3. `⟨]]⟩⟨R⟩(_'text'⟨]⟩M)`

   Make sure that the end block mark is set. Place "text" at the end of each line. Start with the line the cursor is on and stop at the end of the block.

4. `⟨R⟩(3(F)E'text'E)`

   Insert "text" before every third occurrence of the *target* string. (Can be stopped by an end block or ^Y.)

5. `O'END;'O'BEGIN'2(O'<<'40('*')'>>')MO' PROCEDURE'MO2(J)`

   Insert an SPL procedure head above cursor.

6. \[⟨R⟩(⟨R⟩(\L)⟨]⟩M)

Justify a block of text. Note that \L has a special feature which forces a repeat group exit if no text is linked up.

7. ⟨[]⟩\[⟨R⟩(\VI⟨]⟩)

If both blocks are not set, issue a message. Downshift a block of text.

6.7 TERMCAP OPTION

Supported Terminals

Any Terminal that has line insert and line delete capabilities is a candidate for using TESS through the TERMCAP file.

TERMCAP Description

TERMCAP is a file that describes a Terminal. The capabilities are described by listing necessary and optional Terminal functions followed by the specific codes that perform that function. Initialization sequences may also be defined. TESS operation will depend on the capabilities of the Terminal being used. All tess operations are handled in TERMCAP with the exception of function key editing.

The TERMCAP concept was originally designed in the UNIX[1] community. This version, while borrowing much of the UNIX TERMCAP command structure, is not fully compatible with standard UNIX TERMCAP syntax. One major difference is that only one Terminal definition may be included per TERMCAP file. Further, TERMCAP entries are similar but not equivalent.

After the TERMCAP file is designed, it may be invoked with a "FILE" equation. If TESS cannot open the TERMCAP file, then the normal HP Terminal operation proceeds. When a TERMCAP file is present, the entries are read into program memory and the TERMCAP file is closed. TESS uses these entries for all output to the Terminal until program end.

Creating and Activating TERMCAP

The best way to create a TERMCAP file is by altering one that already exists. The record length of the file should be 80 characters wide. To activate the TERMCAP, a file equation must be set at the Terminal. The recommended method to do this is by inserting a file equation into the user-level UDC. The formal designation is:

:FILE TERMCAP=TERMCAPNAME.TERMCAP.CCSCSOFT

TESS will initially read in the capability list from the TERMCAP file and send the initialization string to the Terminal. Some error checking is done at this time. Editing then proceeds as usual. When an entry is omitted, TESS assumes either a default or the null condition for that entry. In the null condition, the visual part of the capability is nullified but the action will still occur in memory. This means that omitting the line deletion string will not disable line deletion in memory.

Writing TERMCAP Entries

Entries in TERMCAP are separated by a colon (:). The first entry gives all of the Terminal's aliases, separated by the broken vertical bar (|). The first name is 2 characters long. The second name is the abbreviation for the Terminal while successive names are fully identifiable names.

The capabilities begin after the alias entry. A colon-backslash-blank (:\ ) indicates the end of an entry and the continuation of the list onto the next record. The next record must begin with a colon (:). Entries may be made one on a line or many on a line. Comments may be made anywhere on the line after the continuation sequence. The last entry ends with a colon (:) and no backslash.

Entry Mnemonics

Each entry begins with a two-character mnemonic. There are three types of mnemonics: Boolean, Numeric, and String. Boolean entries indicate that the Terminal has some sort of feature. They are followed by "=TRUE" or "=FALSE". Numeric capabilities give size parameters. They are followed by "=#" and then a decimal number. String entries hold the code sequence to operate the Terminal. They are followed by "=" and then the string of codes using TERMCAP representations. All entries are delimited by a colon (:).

By referring to the Terminal's user manual, the correct editing codes may be found and assigned to a mnemonic. For instance, the delete character function may be "escape P". The mnemonic "DC" is used for "delete character", so the string would look like ":DC=\EP:\".

While on the subject of character deletion, it should be noted that not all Terminals have this feature. Nevertheless, TESS will allow character deletion by rewriting the line, but to turn this feature on, the "move row left" mnemonic "RL" needs to be set false. Similarly, the "move row right" entry "RR" will probably also need to be set false to allow character inserts.

Entry Representations

The following representations are used to express numbers and functions in the TERMCAP entry. There are no separators other than these elements in TERMCAP syntax:

:	entry separator
#n	decimal number where n≥0 and n≤255
%	entry argument prefix **(see cursor addressing)**
=	assignment
FALSE	capability off **(boolean entries only)**
TRUE	capability on **(boolean entries only)**
\E	escape
\^	actual character (^)
\\	actual character (\)
\#	actual character (#)
\%	actual character (%)
\=	actual character (=)
\nnn	octal number where nnn≥000 and nnn≤377 ie. \072 is a colon (:)
^c	control character **(in this case a control C)**

Terminal Set-up

Due to the differences in Terminal set-up, it is left up to the user to decide exactly what Terminal settings should be programmatically initialized. Some Terminals may not need any initializing at all. There are three places, during a TESS session that initialization may take place. First there is the primary initialization **(IS)** that occurs before any output to the Terminal. Second there is the initialization for character mode editing **(VS)**. The third place **(VE)** is before program termination or when TESS editing suspends to perform line-oriented tasks ie. MPE commands.

What is not possible to set through the initialization string should be attempted manually before program execution. Some typical manual changes might be: data communications configuration, character mode on, auto line feed off, memory lock off, display functions off, set margins, set tabs, and USA ASCII character set. If these functions may be set programmatically, then they may be inserted in either the **(IS)** or the **(VS)** string. When put into the **(IS)** string, they will be performed only once at program start. When put into the **(VS)** string, they will be exercised every time the screen is refreshed. Generally, it is only necessary to make these settings once **(IS)**.

There are two exceptions to this generality. The first is wrap-around lines. During character editing, automatic wrap-around lines should be disabled. If this can not be disabled in any way, **(DW)** should be set to false. If the wrapping capability may be toggled programmatically, then the initialization strings **(VS)** and **(VE)** are where the settings should be placed. Wrap off should be in **(VS)** while wrap on should be in **(VE)**. If wrap around lines can always be disabled **(disabled manually)**, then set **(DW)** true.

The second exception concerns handshaking. MPE needs to know when a data transfer has completed before it may send more. As long as the Terminal is equiped with ENQ/ACK logic, there is no problem. If the Terminal does not support ENQ/ACK, then **(DH)** disable handshaking should be set "TRUE". Generally speaking, if handshaking is disabled, then delays must be included in certain commands to give them enough time to complete **(e.g. cursor addressing)**. This delay function is accomplished by placing a padding factor directly after the equals sign of the mnemonic command **e.g. (CM=6\E&a%r%dc%dR)**. Some experimenting may be required to find the proper amount of padding to include. A good rule of thumb is that as you increase the baud rate, you should also increase any necessary padding.

Cursor Addressing

Cursor addressing is described by the cursor movement entry **(CM)** but row and column positions must be inserted within the string at run time. Therefore, row and column positions are "arguments" to the **(CM)** entry. These arguments are inserted according to the following representations:

%d	zero based decimal number ("0"-"255")
%2	zero based double digit ("00"-"99")
%3	zero based triple digit ("000"-"255")
%.	one byte binary encoded
%+x	add x to value then "%."
%>xy	if value>x add y then "%."
%r	reverse order of row/column **(not sent to Terminal)**
%i	increment row/column **(not sent to Terminal)**
%n	XOR row/column with 0140

For example, the ANSI standard for cursor movement is:

```
:CM=\E[%i%d;%dH:\
     ↑ escape
        ↑ base one row and column
          ↑ row digits
            ↑ ;
             ↑ column digits
               ↑ H
```

Display Enhancement

The first enhancement entry to fill is the stand-out mode switch (SM). "TRUE" means that the Terminal will display enhance by turning the enhancement off at the end of the field and then on at the beginning of the field. "FALSE" means that the Terminal requires the field to be turned on at the beginning, characters or spaces typed, and then the enhancement is turned off.

The next entry to fill is the stand-out glitch number (SG). There are Terminals that use physical space on the screen for the stand-out codes. The amount of space taken by a single enhancement code is called the stand-out glitch. Usually, if the glitch exists, it will be 1 character in length.

After (SF) and (SG) are entered, the end stand-out, full and half inverse video entries (**SE**), (**SO**), and (**SH**) are needed. If half stand-out is not available, full stand out may be used in both (**SO**) and (**SH**).

Memory Handling

TESS requires at least one screen of memory. When only one screen is available, the screen will be re-drawn instead of flipping to the next page. Flipping happens whenever there is dialogue other than editing, ie. the help command. The editor is in "visual" or "character" mode when on the first screen and in "line" mode on the second screen.

If the terminal does not scroll memory above the screen and/or below the screen, the entries (**DA**) and (**DB**) need to be set accordingly. For example, if an insert line deletes the last line off of the screen rather than scroll it down into memory, (**DB**) is set "FALSE". And, if a cursor movement to the last line of the 2nd page of memory is not possible then (**DA**) is set "FALSE".

TERMCAP Commands

The complete set of entry mnemonics is listed below along with the data type of the entry and an example appropriate for the HP-2622 Terminal. For the purpose of documentation, the mnemonics are grouped into four divisions - Editing Functions, Cursor Movement, Terminal Characteristics, and Initializations.

An asterisk (*) after the code indicates that the entry is for TESS only and is not a UNIX TERMCAP standard. The "P" column indicates codes which may require some padding if used without handshaking.

The first line of the file does not fall into any of the four divisions. It is used for identification.

TERMCAP Categories

Identification	Code	P	Type	Entry	
			id	H1|HP2622:\	
Editing Functions					
insert line	AL	p	str	:AL=\EL:\	
clear display	CD	p	str	:CD=\EJ:\	
clear to end of line	CE	p	str	:CE=\EK:\	
delete character	DC	p	str	:DC=\EP:\	
delete line	DL	p	str	:DL=\EM:\	
end insert char mode	EI	-	str	:EI=\ER:\	
insert char mode	IM	-	str	:IM=\EQ:\	
(DC) moves row left	RL*	-	bool	:RL=TRUE:\	**(default=true)**
(IM) moves row right	RR*	-	bool	:RR=TRUE:\	**(default=true)**
Cursor Movement					
back space	BS	-	str	:BS=#8:\	
cursor movement	CM	p	str	:CM=\E&a%r%dc%dR:\	
carriage return	CR	p	str	:CR=#13:\	
home cursor	HO	p	str	:HO=\EH:\	
line feed	LF	-	str	:LF=#10:\	
cursor lower left	LL	-	str	:LL=\E&a0c23R:\	
cursor right one char	ND	-	str	:ND=\EC:\	
cursor up one line	UP	-	str	:UP=\EA:\	