TEDI: a Text EDItor

CTSS, VMS, UNIX, and IBM PC Versions

Clair W. Nielson
Los Alamos National Laboratory
Los Alamos, New Mexico 87545
June 1, 1988

TEDI is a general purpose text editor designed to be used in a variety of applications, including program development and word processing. All versions contain line-oriented commands which may be executed on all kinds of terminals, including hard-copy terminals. Among the advanced line-editing features in TEDI are string and numeric registers, iteration commands, and command files; these features make possible editing programs which can perform complex editing operations. A powerful metacharacter manipulation capability is also provided.

Screen editing for selected terminals is provided in the VMS, UNIX, and IBM PC versions. Screen editing is generally more convenient for entering new text or programs, and for making changes which involve several changes in a localized region. The line editing commands are often more convenient for making global changes and for quickly finding all occurrences of specified patterns.

GETTING STARTED

TEDI is a public file on all Los Alamos CTSS computers and therefore no special arrangements are necessary for access on CTSS. Many system managers have made it public on their UNIX or VMS machines. If this is not the case, you may obtain a copy from the Common File System for your own use. The IBM PC version may also be obtained from CFS.

The latest executables are stored in CFS respectively as /tedi/ctss/tedi, /tedi/unicos/tedi, /tedi/vms/tedi.exe, /tedi/unix/tedi, /tedi/sun/tedi, and /tedi/ibmpc/tedi.exe. All of these may be used after getting them from CFS without any conversion except for the IBM PC version. It is stored in a form suitable for downloading with UNIX KERMIT using the "set file type binary" option. Another file, /tedi/vms/tedi.ibm, is in a form for downloading from VMS KERMIT using the same option. (In order for the VMS KERMIT to work properly, /tedi/vms/tedi.ibm must be obtained with the CFS utility, not with the older MASS utility.)

On VMS it is convenient to define TEDI as a foreign command. In addition, VMS TEDI will start up faster if "installed" by the system manager.

Execution is begun with the command line

    TEDI filename <R>

    or

TEDI <R>

If TEDI has not been defined as a command on VMS, execution may begin with "RUN TEDI" if TEDI.EXE is in your default directory. On VMS you may define your own TEDI command with the definition (presumably saved in your LOGIN.COM file)

TEDI:==$(dev):[(dir)]TEDI

where (dev) and (dir) are the device and directory in which TEDI.EXE resides.

The symbol "<R>" means return.

In the cases where the filename does not appear on the execute line, the most recently edited file will be assumed. This file name and other parameter settings are saved in a file "TEDI.INI" which is in your home directory or in the root directory of the current drive. If this file does not exist, you will be asked for a filename with a prompt. Successful opening of a file will be evidenced by a message giving the filename and the number of lines found in the file. For a new file, the number of lines will be zero.

Any time that TEDI has typed out the command prompt "*" (asterisk) you may give any command, including one of the termination commands "END" or "QUIT". If you have made no changes to the file (for example, you have just typed out selected lines) neither command causes rewriting of any files. However, if you **have** made changes, "END" causes a new version of the file to be written into your filespace, whereas "QUIT" does not, even though changes may have been made. The QUIT exit is appropriate if the changes were erroneous. The END command saves part of the working file so that switch settings may be restored for the next execution. The QUIT command does not save switch settings and future executions resume with default values. If TEDI is aborted with the break key on CTSS or CTRL-C on TOPS-10 or VMS, changes in switch settings will not be saved since the "tedi.ini" file has not been written. If the abort is accidental, editing may be resumed with the "continue" command on VMS and by executing the drop-file on CTSS.

The lines in files edited by TEDI are limited to a maximum length of 150 characters, which may include any combination of upper and lower case. TEDI is always receptive to either case and single case input must be controlled through the use of locking keys on the terminal keyboard. While using TEDI, either the system managed "CTRL-X" (LTSS/CTSS) or "DEL" (TOPS-10/VMS), or the internally managed "CTRL-H" (which echoes as a backspace) may be used for character erase. The line erase is "CTRL-Y" (CTSS), "CTRL-U" (VMS), or "ESC" (IBM PC). On UNIX these characters are set by parameters in your login file. To abort multiple-line commands use "CTRL-Z" followed by a return (CTSS), "CTRL-Z" alone (VMS, IBM PC), or "CTRL-D" (UNIX). To terminate excessive type-out, enter "CTRL-E, i, return" (CTSS) or "CTRL-O" (VMS). If the type out is from a verification, the update will be completed even though the verification stops.

TWELVE BASIC COMMANDS

TEDI has a large instruction set designed to allow experienced users to carry out complex editing operations with a minimum of effort. However, the following subset of commands will be sufficient for most editing operations and should at least be sufficient for a new user.

Line Replacement

IALm/text/text/.../.	Insert After Line	Add text after line m.
IBLm/text/text/.../.	Insert Before Line	Add text before line m.
DLm,n	Delete Lines	Delete Lines m through n.
RLm,n/text/text/.../.	Replace Lines	Replace m through n with text.
CAi,j,m	Copy After	Copy lines i through j after line m.

Pattern Replacement

RPm,n/pat/newpat	Replace Pattern	Replace pattern in lines m through n.
DPm,n/pat	Delete Pattern	Delete pattern in lines m through n.

Type and Search

Tm,n	Type	Type lines m through n.
TAm,k	Type After	Type k lines after line m.
TPm,n/pat	Type Pattern	Type lines from m to n with pattern.
TPA/pat	Type Pattern After	Type next line having Pattern.
TPB/pat	Type Pattern Before	Type previous line with pattern.

In the representations of commands and their arguments above, the slash represents either a return, in which case the editor will prompt for additional input with the prompt shown in quotations, or a logical line feed character (for which the slash itself is suitable if no actual slashes are to be entered as text or pattern), in which case the additional input is typed in on the same line with the command.

The following represent examples of these commands. Remember that these commands must be entered immediately after an asterisk "*" prompt. If any other prompt is given, something other than a command is called for as will be described shortly. The examples show the variation in punctuation possible for numeric line-number arguments. The examples also reveal that commands may be entered in either upper or lower case.

*T 1 5 <R>	(Type out lines 1 through 5)
*dl10,11 <R>	(Delete lines 10 and 11)
*rp10/x/y	(Replace x's with y's in line 10)
*IAL 3 <R>	(Enter text mode to insert new text after line 3)
tp/Chapter	(Type out all occurrences of "Chapter")
*END <R>	(Terminate making a new version of the file)
*quit <R>	(Terminate discarding any changes)

COMMAND FORMAT

Commands consist of a string of letters followed by numeric arguments. Most of the time the letters are abbreviations for the command name. The "END" and "QUIT" commands are exceptional in that the entire command word is typed in. Examples of command abbreviations are "T" for Type, "IAL" for Insert After Line, "RP" for Replace Pattern, and "DL" for Delete Lines. Up to four numeric arguments follow the command abbreviation. The numeric arguments may in general represent a variety of things, but most often the first two numbers represent the first and last line numbers of a range of lines to which the operation is to be applied while the third and fourth represent the beginning and ending column numbers between which pattern matches are to be considered. For example, "RP15,20,1,7" replaces a pattern in lines 15 through 20 if it occurs in columns 1 through 7. Other uses of these numeric arguments are described in the detailed command descriptions. Note that among the basic dozen commands, the Copy After command, CAi,j,m uses the first two arguments as source line numbers and the third as a destination line number. The punctuation for the numeric arguments is flexible. The first number may come immediately after the command abbreviation without a space, or it may be separated from the abbreviation by one space. The subsequent numbers may be separated by either spaces or commas.

Defaults apply to all numeric arguments and are invoked by simply omitting the argument. To default an early argument while specifying a later argument explicitly, successive commas are used. The default for the line number arguments is the current line, which is the last line on which a change or type-out occurred. (The exception is the source line range in file copy and write commands CFA, CFB, WF, W, and WR which default to the entire file.) The default column range is one through 150, i.e., the entire line. The default for count arguments such as the number of lines k in the Type After command, "TAm,k", is one line.

LINE NUMBERS

As was stated in the introductory remarks, TEDI uses implicit line numbers which denote the current position of a line in the file. These numbers run 1, 2, 3, ...,], where "]" is a symbol representing the last line in the file. (This symbol, the right square bracket, may be used in place of a number in a line number position in a command.) One of the properties of such a line number is that it changes if any lines before it are inserted or deleted. These line numbers, therefore, do not acquire permanent significance in the way that line numbers do in some line-oriented editors. While the constant changing of line numbers may be inconvenient in some instances, it is simpler not to be concerned about whether the increment is suitable, and it completely avoids difficulties associated with copying or moving blocks of lines with fixed line numbers. One trick to keep in mind is that if a sequence of changes is being made from a single reference listing, when the changes are made in reverse order from the largest line number to the smallest, the line numbers remain valid until all the changes are complete. However, usually this is not a significant issue since context related information will be used as the principal means of location.

SPECIAL INPUT CHARACTERS

A single return when in command mode causes the next line to be typed. CTRL-Z return types the previous line. A blank followed by a return types the next ten lines, two blanks followed by a return types the next twenty lines, and three blanks followed by a return types the next thirty lines. In any line number argument, a negative number is interpreted relative to the current line. In place of numeric arguments, one may use "." to reference the current line, "]" to reference the last line in the file, and "*" to represent "a very large number". These symbols and numeric arguments may be joined by plus and minus signs. Thus

    T.-5,.+5

types from five lines before the current line to five lines after the current line,

    TA,*

types the rest of the file starting with the next line, and

    T]-15,]

types the last sixteen lines in the file.

A command (including its numeric arguments) and any associated strings must be separated from each other by either returns, in which case TEDI will prompt for the string arguments, or by a dynamically determined logical line feed character, in which case the entire sequence may be entered on one line. The logical line feed is a separator such as ";", "/", or "\" which is given immediately after the first command on a line and then applies for the duration of that command (perhaps over many lines for multiple line commands). For example, the multiple line version for a "Type Pattern" is

*TP <R>
P:Hello <R>

whereas the single line version is

*TP/Hello <R>

using the logical line feed character "/". The **P:** in the first case is a prompt given by TEDI. Any non-alphanumeric except *, .,], +, -, or blank may be used as the logical line feed character. In addition, the ASCII escape character **always** functions as a logical line feed, even when another logical line feed character is in use. However, to insure system independence, the use of the escape for this purpose is not recommended.

One may wish to input into a text line or string the logical line feed character itself. To make this possible, the force literal character, the grave accent "‘", is used. If it precedes a special character, the special character is input into the text or string regardless of whether or not it is currently the logical line feed (or the tab character, see the STC command). To input the grave accent itself, type two of them "‘‘". (In many cases one may simply chose to use an alternate logical line feed character and avoid the need for the "‘" sequence.) In addition to its force literal function, the grave accent is used to input control characters. The grave accent followed by an upper-case alphabetic character inputs the associated control character. For example, "‘C" inputs "CTRL-C" into the text or string.

TEXT INPUT MODE

Two of the examples above, IAL, Insert After Line, and RL, Replace Lines, cause entry into text entry mode. This means that the editor is prepared to accept lines of text to be entered into the file. A prompt of the ampersand "&" is given for each line, but the ampersand does not go into the line. (The prompt may be turned of--see the section called switch settings--but it is helpful as a reminder of the fact that the editor is in text entry mode.) When you have entered all of the lines you wish, enter one line with only a period "." at the beginning of the line and then type a return. The period terminates text mode and returns the editor to command mode. (Although it will be rare to wish to enter a line with only a period in the first position, it can be done by typing one or more blanks after it. Since before writing the line to disk TEDI removes all trailing blanks, the line that enters the file contains only the period in the first position. The extra blanks on type-in tell the editor that you do not wish to terminate text mode.) To enter text mode for a new file, the line number is simply omitted from the insert command "IAL". When you leave text mode by this means, the editor will immediately begin typing out the consequences of the change. This is called verification. (Verification may be turned off--see section called "Settings".)

Example with multiple line input:

*IAL <R>
&Now is the time <R>
&for all good men. <R>
&.

Example with single line input:

*IAL/Now is the time/for all good men/. <R>

COLUMN REFERENCING

Column referencing is possible through the third and fourth arguments of ordinary string matching commands, or though explicit column referencing commands. An example of the former is "TS*,,1,7/10" which will find the symbol "10" only if it occurs in columns one through seven. An example of the latter is "RC,,6;2" which places a digit "2" in column 6 of the current line, regardless of the current contents of column 6.

Default cases of the column referencing commands are of frequent use. The AC "After Columns" command with no explicit column number puts the string after the last non-blank character in the line. The BC "Before Columns" command with no explicit column number places the string before column one, i.e., at the beginning of the line. The DAC "Delete After Columns" command with no explicit column reference deletes all trailing blanks. (TEDI always deletes trailing blanks when creating text; however, files converted from some other origin sometimes have wasteful trailing blanks.)

PATTERNS, SYMBOLS, AND METATEXT

Certain commands search for the occurrence of specified strings of characters. These strings are limited to 80 characters, although the lines that result from changes using the strings may be up to 150 characters long. A pattern is any string of legal characters. A symbol is a pattern that is delimited in the text by special characters or blanks. This distinction is useful as a means of changing variables in a source program or full words in a text file. For example, it allows finding all occurrences of the FORTRAN variable "I" without interference from occurrences of the letter "I" in the symbol "PI" or the word "IF". Pattern commands use the letter "P" to denote pattern, i.e., "TP" for Type Pattern, while symbol commands use the letter "S", i.e., "RS" for Replace Symbol.

For more powerful string matching and replacement, TEDI defines several metatext (wild card) constructs. A metatext string is a string taken to stand for more than the literal sequence of characters constituting it by defining some special characters to be given generalized meanings. To indicate that the referenced strings are to be so interpreted, the letter "X" is used in the commands; in the command descriptions, the abbreviation "mtext" is used for metatext in a matching context while "rtext" is used for metatext in a replacement context.

For metatext matching purposes, the case is not considered. Thus one use of metatext is to search for strings whose case is unknown. If one of the special characters introduced below is desired literally, it should be preceded in the mtext string by the AT sign "@". The AT sign itself may be specified in a metatext string as "@@". Finally, note that metatext searching is much slower than either pattern or symbol searching, so such a search should be used only when its generality is needed.

(1) Single Wild Characters: "?" matches any character, "&" matches any alphanumeric, and "!" matches any non-alphanumeric. No mechanism for identification of these single characters is available for use in replacement strings, but the same result may be obtained by using a character class of length one using the notation described in the next paragraph.

Example:

```
TX*,,72,80;?
```

types all lines containing anything in columns 72 through 80.

(2) Character Classes: Brackets enclosing a string of characters matches any one of the enclosed characters. For example [xyz] matches a single x, a single y, or a single z. [&$_] matches any single alphanumeric, dollar, or underline. The minus sign between two characters indicates a range of characters. For example [I-N] matches any character in the range from I to N inclusive. In replacement strings, use [] without any enclosed characters to obtain whatever was matched in the same character class, same meaning ordinal position from left to right. To change the order, enclose the ordinal in the brackets in the replacement string. For example, [3] enters into the replacement string whatever was matched by the third character class in the match string.

Example:

```
RX*;5[0-9];8[]
```

replaces each two digit number beginning with 5 by a two digit number beginning with 8 and ending in the same digit as the original.

(3) Ellipses: For matching variable length strings, the dot-dot-dot notation is used. "..." matches anything. For example "m...e" matches both "message" and "mad ape". "..&" matches any number of contiguous alphanumeric characters. "..!" matches any number of contiguous non-alphanumeric characters. (The ellipses "..?" is identical to "..."; the latter is introduced because it is easily typed and corresponds to common usage.) "..a" matches any number of contiguous a's. "..[xyz]" matches any strings composed solely of x, y, and z. Note that zero occurrences is acceptable, so without the presence of something else an isolated ellipsis will match every line, using zero of the quantities as a successful match. In replacement strings the simple ellipsis is used to represent any of the special forms. The correspondence is made in an ordinal manner from left to right. To change the order, use the form "..n" where n is the ordinal. For example, in a replacement text string, "..3" invokes whatever was matched by the third ellipsis in the match string, regardless of the type of that ellipsis. In all cases, the shortest possible ellipsis match is taken.

Examples:

```
RX*;PRINT...,;WRITE (5,...)
```

replaces all FORTRAN PRINT statements with WRITE statements with logical unit 5 and the same format number while

RX*;WRITE...(5,...);PRINT ..2,
reverses the change.

(4) Symbols: Metatext strings enclosed in angle brackets match only if preceded and followed by either a separator or by the beginning or the end of the line. Thus <x..&> matches all symbols beginning with x. A single angle bracket may be used to obtain a one-sided symbol.

Example:

TX*;<x[1-9]>

types out all lines containing the **symbol** x1, x2, ..., or x9.

TAB STOPS AND INDENTATION

The automatic insertion of blanks before specified points in insertion or replacement text is provided by the Set Tab Stops "STS" and Set Tab Character "STC" commands, by the Replace Data "RD" command, and by the Set Indent "SI" command.

Both left adjust and right adjust tabs are provided. A left adjust tab stop is specified by a positive or unsigned number and causes the following text to be preceded by enough blanks to cause the next character to begin in the column number equal to the value of the tab stop. Columns start at one. A right adjust tab stop is specified by a negative number. The text following the tab character is scanned for the next blank or tab character. If the text is not already too far right, the text is preceded by enough blanks to cause the last character in the word to be placed one column to the left of the column number equal to the absolute value of the tab stop. The tab character itself is user selectable with the Set Tab Character "STC" command. In most cases one occurrence of the current tab character causes the action described in the preceding paragraph to take place. However, if the tab character is a blank, **two** blanks cause the tabbing to take place. The default tab character **is** the blank which is very easy to type. Left and right adjust tab stops may be combined in the STS command.

The above tabbing is in effect for text input lines, i.e., those entered after the insert and replace line commands "IAL", "IBL", and "RL". No special interpretation of the tab character is made by the pattern replacement commands. However, it may be desirable to replace a pattern while retaining the tab stop alignment. To this end, a special pattern replacement command, Replace Data "RD" is provided. The columnar range of its action is specified by considering the line to be divided into data fields which extend between successive tab stops. Specifically, the d'th data field extends from (|ts(d-1)|) to (|ts(d)|-1) with the understanding that the first data field begins at column one. The command Replace Data, "RDm,n,d/pat" will replace the d'th data field with the replacement pattern. For example, the tab stops set by "STS 10,20,35,72" would be suitable for entering a COMPASS assembly language program on LTSS, separating the label, operation, operand, and comment fields. To replace only the comment field, the command "RD,,4/new comment" would replace the comment while retaining its proper position on the line.

The Set Indent "SI i,j" command provides a more dynamic columnar positioning and is suitable for emphasizing the logical structure of computer programs. The initial indent level is set to column i and the increment to column j. Only text input lines are affected when the SI command has been given and to remind the user, they are prompted by the special prompt ">". When this has been done, any lines entered in text mode will be indented to the current level with the following exceptions. If a line begins with a period, the text following will begin in column one. If a line begins with a plus sign, the indent level will be incremented by j before beginning the following line, and a minus will similarly decrement the indent level. A zero will reset the indent level to i. A colon as the first character will cause all characters through the first blank to be placed starting at column two, and then the remainder starting at the current indent level (this is for statement labels in many languages) while a comma behaves similarly except the first string starts at column i-1 as is appropriate for a FORTRAN continuation. The indent function may be disabled by an SI command with no arguments. The indent parameters i and j are saved across executions in the working file, but the current indent level is not and is restarted at the beginning of each execution. As an example of using the Set Indent command to input RATFOR text, the input string

```
SI3,3;IAL;while (getlin(buffer,STDIN) ^= EOF);+[
if (buffer(1) == EOS);+next;-else;+call putlin(buffer,STDOUT);-];.
```

causes the text to be entered as

```
  while (getlin(buffer,STDIN) ^= EOF)
     [
     if (buffer(1) == EOS)
        next
     else
        call putlin(buffer,STDOUT)
     ]
```

Multiple occurrences of the plus or minus indicators are properly interpreted.

REGISTERS

Ten numeric registers, R0, R1, ..., R9, may be used for temporary storage of integer data. The command Set Register, "SRn,k", sets register n to value k. The register may then be used in place of a numeric argument in a command. For example, "SR3,5" followed by "T R3" types line 5. Note that the Set Register command is a single mnemonic so there is no space between the "S" and the "R" whereas when the register is used to represent a number in the Type command there is a space between the "T" and the "R". The greatest utility of these registers is in DO loops and command files, but manual use is helpful when doing copies or moves in long files as a means of recording line numbers. As a DO-LOOP example,

"SR1,1/DOP*/ END/SR2,./TD R1,R2/SR1,R2+1/NDO"

will type out all FORTRAN program units in a delimited form such that each function or subroutine begins on a new page. Future versions of TEDI will set R0, R8, and R9 after every string match command, so only registers R1 through R7 are recommended for general use in command files. Type commands useful with registers are "TI", Type Immediate, and "TSI", Type String Immediate. The following example counts and displays the number of lines in which the symbol X1 occurs.

"SR1,0;DOS*;X1;SR1,R1+1;NDO;TSI;Number of X1's =;TI R1"

REMEMBERED STRINGS

Three different strings are remembered. They are respectively the match string, the replacement string, and the do string. These strings contain the string most recently used to reference them. If in any command the null string is input for one of the string arguments, the remembered value is invoked. For example, the command sequence

TPA;Hello;RP;;Goodbye

will replace the next "Hello" found with "Goodbye" without the need to type "Hello" twice. One of the consequences of remembered strings is that the replace commands may not be used with null input to delete strings--the delete commands must be used instead.

In order to explicitly set the match, replacement, and do string buffers from the contents of a line, the set string commands are used. They are SSMm,,c1,c2", "SSRm,,c1,c2", and SSDm,,c1,c2. These set the appropriate string with the contents of line m, columns c1 through c2. Trailing blanks will be supplied if necessary. As an example, to move columns 21 through 30 to the beginning of lines 100 through 200, use the following command:

DO100,200/SSR,,21,30/DC,,21,30/BC//NDO

KEY-EDIT SCREEN EDITING (VMS, UNIX, and IBM PC)

A screen editor for VT-220 and VT-100 terminals and their emulators, and for the IBM PC, is invoked with the "K", Key Edit command. The command "K" or "K m" invokes the screen editor with the initial window determined by the line number "m". The Key-edit command is terminated and line editing resumed with the CTRL-Z key.

When in key-edit mode, the cursor keys move the cursor to any point on the displayed screen, and cause scrolling of the screen at its boundaries. Other key edit functions may be invoked either with control key combinations in which an alphabetic letter is struck while the control key is held down. On most terminals and workstations, alternative single-stroke function keys may be used instead of the control key combinations.

In the following descriptions, the column marked ALL shows the keys one can use on any ANSII video terminal, while the columns marked VT220, IBM PC, and VT100 show the alternate single-key forms available on these terminals or PC's. When using an emulator, the mapping of VT-220 or VT-100 function keys must be determined from the emulator documentation.

ALL	ACTION	VT220	IBM PC	VT100
CTRL-Z	Leave key edit and enter line edit; to return to key-edit type K <ret> or K m <ret> (m is a line number)			
CTRL-?	Help. Display brief help screen (CTRL-MINUS on some emulators)	Help	F1	
Arrows	Move about the text			
CTRL-H	Scroll ten lines up	Prev Screen	Page Up	PF1 or Back Space
CTRL-J	Scroll ten lines down	Next Screen	Page Down	PF2 or Line Feed
CTRL-E	Jump between first and last characters of line	F14	End	
CTRL-F	Jump between first and last lines on screen	F12	Home	
CTRL-L	Locate. Prompts for pattern. Searches forward if terminated with Enter or CTRL-J or Down Arrow; backwards if terminated with CTRL-H or Up Arrow; CTRL-L repeats without other characters cycles pattern, symbol, and metatext search.	Find	F2	
Enter or Return	Causes a blank line to be inserted after the present line			
CTRL-T	Increase (soft Tab) indent level, default three columns. First CTRL-T moves to current indent level, use with CTRL-E	F8	F4	
CTRL-R	Decrease (Reverse tab) indent level, default three columns	F7	F3	

CTRL-P	Push. Enters insert mode. A single CTRL-P puts the editor in volatile insert mode ("i displayed"); overwrite mode is entered on move to new line. Two CTRL-P's in a row lock the editor in insert mode ("I displayed"); overwrite mode is restored only after another CTRL-P	F9	Insert	PF3
CTRL-D	Delete the character under the cursor	F14	Delete	PF4
DEL or <-	Delete character before the cursor			
CTRL-K	Delete from cursor to end of line or, if at beginning of line, delete entire line; replaces buffer	Remove	F7	
CTRL-U	Delete everything before cursor on present line		ESC	
CTRL-N	JoiNs the present line to the next line	F17	F11	
CTRL-V	DiVides the line before the present cursor position	F18	F12	
CTRL-G	Get. Enters select mode where patterns or lines will be marked for deletion or copying. A single CTRL-G puts editor in pattern select mode ("P" displayed). An second CTRL-G or Up Arrow, Down Arrow, Page Up or Page Down activates line select mode ("L" displayed). The arrow and scroll keys are used to complete the selection	Select	F8	
CTRL-K or CTRL-D	When in select mode, deletes high-lighted characters, puts them in buffer, and ends select	Remove	F7	
CTRL-G	When in select mode, the final CTRL-G puts highlighted text in pattern or line buffer without deletion and ends select mode	Select	F8	

CTRL-A or CTRL-B	When in select mode, adds high-lighted material to that already in buffer and ends select	Insert Here	F6
CTRL-A	When not in select mode adds contents of buffer as lines After current line	Insert Here	F6
CTRL-B	When not in select mode adds contents of buffer as pattern Before cursor position	F13	F5
CTRL-]	FILL lines from present cursor position to the next line that is either all blank, begins with a blank, or begins with a period	F10	F9
CTRL-W	WORLD command which prompts on status line for letter: W Write (save file) E End saving file Q Quit not saving file O Open new file R Repaint screen S Show file name	DO	F10

TEDI saves both line-edit and key-edit status information, as well as the most recently used file name, in a file "TEDI.INI" in the home directory. This file allows restoration of parameters each time you start up TEDI. Starting TEDI without a file name uses the saved name.

There are two line edit commands, "SK" set key edit, and "NK" no key edit, which cause the editor to start up in either key edit or line edit mode. The default indents for the CTRL-T and CTRL-R soft tabs can be altered by the line edit "SI" command.

On VMS the editor reads the page length value from the VMS terminal settings. Users of the Visual 550 should SET TERM/PAGE=33 while users of the Tektronix 4100 and 4200 series should set page length to either 30 or 32 lines. The setting of page length for users of networked DEC workstations is automatic.

COMMAND FILES

Files containing sequences of commands may be executed. The file is invoked with the Use command, "Um1,m2,...,mk/filename". The numeric arguments are optional and cause setting of the registers R1, R2, ..., Rk before beginning execution of the command file. For example, if a sequence of commands is prepared in the file "COMFIL", the commands will be executed as a consequence of the command "U/COMFIL". Command files may also be invoked using the indirect notation "@" either on the initial execute line or in any of the Open commands. For example, "TEDI @COMFIL" or "O;@COMFIL". Lines beginning with an exclamation point "!" in column one are considered comment lines. Leading blanks in commands (although not in strings) are ignored, so that commands may be indented for readability. Command files may not be nested or included in the range of a "DO".

As an example of the use of command files, consider the mailing of form letters to a standard list of addressees. Let the file "letter" contain the text of the letter with a line containing only the word "INSERT" at the point at which the address and salutation should be inserted. Let the file "address" contain the addresses, each of which is followed by the proper form of the recipient's name to be used in the salutation, followed by a line containing only the symbol "EOA" (end of address). Finally, let the file "mail" consist of the following lines:

```
nv/o/temp/dl*
cfa/address/sr1,.
cfa/letter
lpa r1/INSERT/sr2,.
sr3,1/nn
dop 1,r1/EOA/sr4,./tsi/‘L/t r1+1,r2-1/t r3,r4-2/tsi/‘M‘J
tsi/Dear /t r4-1/t r2+1,]/sr3,r4+1/ndo
tsi/‘L/quit
```

The command "tedi @mail" on a form-feed sensitive terminal will cause a copy of the letter to be typed out to each of the addressees.

FILE HANDLING COMMANDS

O;filename Open

Open an existing filename or, if filename does not yet exist, start a new file of that name. If another file is already open when this command is given, and if any changes have been made in it, a new version is written before proceeding with the new filename.

OD;filename Open Discarding

Open an existing or new file as in the Open command. In this case, however, a previously opened file is not updated, even if changes have been made. This is a useful command when erroneous changes have been made and a new copy of the original file is desired.

W Write

Write out a new version of the working file. This command should be used frequently to provide "snapshots" of the editing process and to provide a recovery file to use to recover from editing or hardware failures. The equivalent of a Write is automatic when an End command is given or a new file is opened with the Open command.

WFm,n;filename Write File

Write lines m through n of the working file into a file of the name filename. No change of the name of the working file takes place. The default range is the entire working file.

WFNm,n;filename Write File Numbered

Write lines m through n of the working file into a file of the name filename. Include the line numbers in the file at the beginning of each line. No change to the name of the working file takes place. The default range is the entire working file.

WR;filename Write Renaming

The present working file is written out to filename. Moreover, the name associated with the working file (the name used when a file write is necessary for the End or Open commands) is changed to filename.

Um1,...,mk;filename Use

Use the contents of filename as a source of commands. The file should be created to contain commands identical to those that would otherwise be typed in at the terminal. At the end of reading through filename and executing its commands, TEDI returns to the terminal for additional commands. The numeric arguments m1,...,mk are optional and cause setting of the registers R1, R2, ..., Rk before beginning execution of the command file. This allows passing "arguments" to the command file in the manner of a subroutine or macro. Command files may not be nested or included in the range of DO commands.

EXCHANGE CURSOR EDIT

Xm,n Exchange

Exchange types out lines m through n one at a time, prompting with a colon beneath the line for the user to specify the exchange of characters desired in the line above.

BLANK -- replicate the character above.

\	BACKSLASH -- delete the character above.
_	UNDERSCORE -- replace the character above with a blank.
^...^	CARET string CARET -- insert the string before the character above the first caret. The ellipsis stands for any character(s) except carets.
|	VERTICAL BAR -- split the line before the character above the bar.

TYPE, PRINT, and LOCATE COMMANDS

The type commands cause type out of the selected lines at the controlling terminal. The print commands send output to a printer attached to an ANSI printer port on the terminal such as that on the VT-100 printer port option. The locate commands cause no actual output. They do, however, carry out the same search procedures as the type and print commands and affect the current line number in the same way. They are useful for locating desired lines in command files, particularly within the range of "DO" iteration loops.

Tm,n	Type
Pm,n	Print
Lm,n	Locate

Type, print, or locate lines m through n.

TAm,k	Type After
PAm,k	Print After
LAm,n	Locate After

Type, print, or locate the k lines after line m.

TBm,k	Type Before
PBm,k	Print Before
LBm,k	Locate Before

Type, print, or locate the next k lines after the current line.

TNk	Type Next
PNk	Print Next
LNk	Locate Next

Type, print, or locate the k lines before line m.

TPm,n,c,d;pat	Type Pattern

PPm,n,c,d;pat	Print Pattern
LPm,n,c,d;pat	Locate Pattern

Type, print, or locate all lines from m through n which contain pat in columns c through d.

TPAm,k,c,d;pat	Type Pattern After
PPAm,k,c,d;pat	Print Pattern After
LPAm,k,c,d;pat	Locate Pattern After

Type the next k lines containing pat in columns c through d. If m is present, start the search after line m.

TPBm,k,c,d;pat	Type Pattern Before
PPBm,k,c,d;pat	Print Pattern Before
LPBm,k,c,d;pat	Locate Pattern Before

Type the previous k lines containing pat in columns c through d. If m is present, start the search before m.

TSm,n,c,d;sym	Type Symbol
PSm,n,c,d;sym	Print Symbol
LSm,n,c,d;sym	Locate Symbol

Type, print, or locate all lines from m through n which contain sym in columns c through d.

TSAm,k,c,d;sym	Type Symbol After
PSAm,k,c,d;sym	Print Symbol After
LSAm,k,c,d;sym	Locate Symbol After

Type, print, or locate the next k lines containing sym in columns c through d. If m is present, start the search after line m.

TSBm,k,c,d;sym	Type Symbol Before
PSBm,k,c,d;sym	Print Symbol Before
LSBm,k,c,d;sym	Locate Symbol Before

Type, print, or locate the previous k lines contain sym in columns c through d. If m is present, start the search before line m.

TXm,n,c,d;mtext	Type TeXt
PXm,n,c,d;mtext	Print TeXt
LXm,n,c,d;mtext	Locate TeXt

Type, print, or locate all lines from m through n which contain mtext in columns c through d.

TXAm,k,c,d;mtext	Type TeXt After
PXAm,k,c,d;mtext	Print TeXt After
LXAm,k,c,d;mtext	Locate Text After

Type, print, or locate the next k lines containing mtext in columns c through d. If m is present, start the search after line m.

TXBm,k,c,d;mtext	Type TeXt Before
PXBm,k,c,d;mtext	Print TeXt Before
LXBm,k,c,d;mtext	Locate TeXt Before

Type, print, or locate the previous k lines containing mtext in columns c through d. If m is present, start the search before line m.

TOm,n	Type Octal
POm,n	Print Octal

Type or print lines m through n displaying each character in its 7-bit octal representation.

TDm,n,k,i	Type Decimal
PDm,n,k,i	Print Decimal

Type or print lines m through n displaying each character in its decimal representation.

TCm,n	Type Columns
PCm,n	Print Columns

Type or print lines m through n preceded by a list of column numbers.

TIk	Type Immediate
PIk	Print Immediate

Type or print the numerical values k. In practice k is usually a register (see discussion of registers above).

TSI;string	Type String Immediate
PSI;string	Print String Immediate

Type or print "string". Useful in command files to send messages.

SWITCH SETTINGS

S Settings

Report the value of all switch settings.

SIi,j Set Indentation

This command initiates a special mode for the text input commands IAL, IBL, and RL to cause systematically indented text so as to emphasize program structure. If i is present and not zero, it establishes the initial indent level and j establishes the increment. If both i and j are absent, indent mode is turned off. The operation of indent mode is explained in the section "Tab Stops and Indentation".

STC;char Set Tab Character

Sets the tab character to char. Subsequent occurrences of char in text input mode (in response to the IAL, IBL, and RL commands) will be expanded to blanks in accordance with the column numbers set in the STS, Set Tab Stops, command. The blank, the initial setting, has a special interpretation when it is the tab character--an even number of blanks is interpreted as half that many tabs while an odd number of blanks is taken literally.

STSc,...,d Set Tab Stops

Set the tab stops to c,...,d up to a maximum of 20. There are no initial settings.

SCC Set Control Characters

This switch causes control characters, other than the final return line-feed, to be printed in the "caret" notation, i.e., a caret followed by an upper case alphabetic character. This switch combined with the force literal character allows fairly complete manipulation of nonprinting control characters in the text.

NCC No Control Characters

This switch disables the "caret" mode of printing control characters and instead transmits them literally to the terminal. This is the initial setting.

SLWn Set Line Width

CTSS only. Causes line wrap on terminals which do not provide it. The default value is 80.

SN Set Numbers

Precede each line of terminal output with its implicit line number. This is the initial setting.

NN No Numbers

Do not precede lines with their line numbers.

SV Set Verification

Type out all changes that occur as a result of line or pattern replacement commands. This is the initial setting.

NV No Verification

Do not type out verification of changes as they are made.

SCTc,d Set Columns Typed

Type out only columns c through d in subsequent type and print commands. A SCT command with no arguments restores the normal situation wherein all columns present are typed.

SXP;char Set TeXt Prompt

When in text entry mode, i.e., when using the IAL, IBL, or RL commands, the input lines are normally prompted with the ampersand. This command allows setting the prompt to some alternative character.

NXP No Text Prompt

This disables the text prompt altogether.

SRi,k Set Register

Set register i to the integer value k.

SSMm,,c,d Set String Match

Set the match string to the contents of line m columns c through d.

SSRm,,c,d Set String Replace

Set the replace string to the contents of line m columns c through d.

SSDm,,c,d Set String Do

Set the DO string to the contents of line m columns c through d.

INSERT COMMANDS

Insert is the default command in the sense that the initial "I" in any of the insert commands may be omitted. For example, "AP", After Pattern, is identical to "IAP", Insert After Pattern.

IALm;text;...;text;. Insert After Line

Insert new lines after line m. Text input is terminated by a line containing only a period in column one. To enter a line containing only a period, enter a blank after the period.

IBLm;text;...;text;. Insert Before Line

Insert new lines before line m. Text input is terminated by a line containing only a period in column one. To enter a line containing only a period, enter a blank after the period.

IAPm,n,c,d;pat;newpat Insert After Pattern

Insert newpat after each occurrence of pat in columns c through d of lines m through n.

IBPm,n,c,d;pat;newpat Insert Before Pattern

Insert newpat before each occurrence of pat in columns c through d of lines m through n.

IASm,n,c,d;sym;newpat Insert After Symbol

Insert newpat after each occurrence of sym in columns c through d of lines m through n.

IBSm,n,c,d;sym;newpat Insert Before Symbol

Insert newpat before each occurrence of sym in columns c through d of lines m through n.

IAXm,n,c,d;mtext;rtext Insert After TeXt

Insert rtext after each occurrence of mtext in columns c through d of lines m through n.

IBXm,n,c,d;mtext;rtext Insert Before TeXt

Insert rtext before each occurrence of mtext in columns c through d of lines m through n.

IACm,n,c;newpat — Insert After Column

Insert newpat after column c in lines m through n. If c is absent, after the last non-blank character.

IBCm,n,c;newpat — Insert Before Column

Insert newpat before column c in lines m through n. If c is absent, insert before column one.

REPLACE COMMANDS

RL;text;...;text;. — Replace Lines

Replace lines m through n with new lines. Text input is terminated by a line containing only a period in column one. To enter a line containing only a period, enter a blank after the period.

RPm,n,c,d;pat;newpat — Replace Pattern

Replace each occurrence of pat in columns c through d of lines m through n with newpat.

RSm,n,c,d;sym;newpat — Replace Symbol

Replace each occurrence of sym in columns c through d of lines m through n with newpat.

RXm,n,c,d;mtext;rtext — Replace TeXt

Replace each occurrence of mtext in columns c through d of lines m through n with rtext.

RCm,n,c,d;newpat — Replace Columns

Replace columns c through d of lines m through n with newpat.

RDm,n,d;newpat — Replace Data

Considering the d'th data field to extend from the d-1'th tab stop to one column before the d'th tab stop, replace the d'th data field with newpat in lines m through n.

RAPm,n,c,d;pat;newpat — Replace After Pattern

Replace everything after the first occurrence of pat in columns c through d of lines m through n with newpat.

RASm,n,c,d;sym;newpat   Replace After Symbol

> Replace everything after the first occurrence of sym in columns c through d of lines m through n with newpat.

RAXm,n,c,d;mtext;rtext   Replace After TeXt

> Replace each occurrence of mtext in columns c through d of lines m through n with rtext.

RACm,n,c;newpat   Replace After Column

> Replace everything after column c in lines m through n with newpat.

RBPm,n,c,d;pat;newpat   Replace Before Pattern

> Replace everything before the first occurrence of pat in columns c through d of lines m through n with newpat.

RBSm,n,c,d;sym;newpat   Replace Before Symbol

> Replace everything before the first occurrence of sym in columns c through d of lines m through n with newpat.

RBXm,n,c,d;mtext;rtext   Replace Before TeXt

> Replace everything before the first occurrence of mtext in columns c through d of lines m through n with rtext.

RBCm,n,c;newpat   Replace Before Column

> Replace everything before column c in lines m through n with newpat.

RLUm,n,c,d   Replace Lower by Upper

> Replace all lower case characters in columns c through d of lines m through n with the corresponding upper case characters.

RULm,n,c,d   Replace Upper by Lower

> Replace all upper case characters in columns c through d of lines m through n with the corresponding lower case characters.

DELETE COMMANDS

DLm,n	Delete Lines

Delete lines m through n.

DPm,n,c,d;pat	Delete Pattern

Delete each occurrence of pat in columns c through d of lines m through n.

DSm,n,c,d;sym	Delete Symbol

Delete each occurrence of sym in columns c through d of lines m through n.

DXm,n,c,d;mtext	Delete TeXt

Delete each occurrence of mtext in columns c through d of lines m through n.

DCm,n,c,d	Delete Columns

Delete columns c through d in each of lines m through n.

DAPm,n,c,d;pat	Delete After Pattern

Delete everything after the first occurrence of pat in columns c through d of lines m through n.

DASm,n,c,d;sym	Delete After Symbol

Delete everything after the first occurrence of sym in columns c through d of lines m through n.

DAXm,n,c,d;mtext	Delete After TeXt

Delete everything after the first occurrence of mtext in columns c through d of lines m through n.

DACm,n,c	Delete After Column

Delete everything after column c in lines m through n. This command also removes trailing blanks.

DBPm,n,c,d;pat	Delete Before Pattern

Delete everything before the first occurrence of pat in columns c through d of lines m through n.

DBSm,n,c,d;sym Delete Before Symbol

Delete everything before the first occurrence of sym in columns c through d of lines m through n.

DBXm,n,c,d;mtext Delete Before TeXt

Delete everything before the first occurrence of mtext in columns c through d of lines m through n.

DBCm,n,c Delete Before Column

Delete everything before column c in each of lines m through n.

MOVE and COPY COMMANDS

MAm,n,s Move After

Move lines m through n after line s.

MBm,n,s Move Before

Move lines m through n before line s.

CAm,n,s Copy After

Copy lines m through n after line s.

CBm,n,s Copy Before

Copy lines m through n before line s.

CFAm,n,s;filename Copy File After

Copy lines m through n of filename after line s of the working file. A nonstandard default of 1 for m and the last line of filename for n is used for this command.

CFBm,n,s;filename Copy File Before

Copy lines m through n of filename before line s of the working file. A nonstandard default of 1 for m and the last line of filename for n is used for this command.

DIVIDE and JOIN COMMANDS

VAPm,n,c,d;pat — DiVide After Pattern

Divide line into new lines after each occurrence of pat in columns c through d of lines m through n.

VASm,n,c,d;sym — DiVide After Symbol

Divide line into new lines after each occurrence of sym in columns c through d of lines m through n.

VAXm,n,c,d;mtext — DiVide After TeXt

Divide line into new lines after each occurrence of mtext in columns c through d of lines m through n.

VACm,n,c — DiVide After Column

Divide line into new lines after column c in each of lines m through n.

VBPm,n,c,d;pat — DiVide Before Pattern

Divide line into new lines before each occurrence of pat in columns c through d of lines m through n.

VBSm,n,c,d;sym — DiVide Before Symbol

Divide line into new lines before each occurrence of sym in columns c through d of lines m through n.

VBXm,n,c,d;mtext — DiVide Before TeXt

Divide line into new lines before each occurrence of mtext in columns c through d of lines m through n.

VBCm,n,c — DiVide Before Column

Divide line into new lines before column c in each of lines m through n.

JLm,n — Join Lines

Join lines m through n into one line.

ITERATION COMMANDS

It is often desirable to execute a command for those lines in a range of lines which satisfy some pattern matching conditions. For simple cases, the "RP", Replace Pattern, command is adequate. However, if the conditional pattern and the pattern to be altered are different, a more general construct is required. TEDI provides "DO" commands for this purpose. A "DO" command is followed by a sequence of additional commands which are to be executed for those lines implied by the "DO". The sequence must be terminated with an "NDO", End Do command. The separate elements of the command sequence may be separated by either logical line-feed characters or with actual returns.

For example, to delete the symbol "X" in every line which also contains the pattern "ABC" one would use the sequence,

```
DOP*;ABC;DS;X;NDO
```

DO's may be nested up to five deep. For example, to delete all lines which contain the symbol "PI" but do not contain the pattern "RADIUS", one would use the sequence,

```
DOS*;PI;DNP;RADIUS;DL;NDO;NDO
```

It is important to note that the first DO has the asterisk following it, indicating "all lines", whereas the second DO does not, indicating its execution only for the "current line", i.e., the line in which a "PI" has been found.

The total character count of a nested sequence of DO commands must be less than or equal to 512, which includes all characters from the outermost DO to the outermost NDO.

DOm,n Do

Repeat the following commands up until the next matching NDO for all lines in the range m through n.

DOPm,n,c,d;pat Do On Pattern

Repeat the following commands up until the next matching NDO for those lines in the range m through n which contain pat in columns c through d.

DOSm,n,c,d;sym Do On Symbol

Repeat the following commands up until the next matching NDO for those lines in the range m through n which contain pat in columns c through d.

DOXm,n,c,d;mtext Do On TeXt

Repeat the following commands up until the next matching NDO for those lines in the range m through n which contain mtext in columns c through d.

DNPm,n,c,d;pat Do No Pattern

Repeat the following commands up until the next matching NDO for those lines in the range m through n which do not contain pat in columns c through d.

DNSm,n,c,d;sym Do No Symbol

Repeat the following commands up until the next matching NDO for those lines in the range m through n which do not contain sym in columns c through d.

DNXm,n,c,d;mtext Do No TeXt

Repeat the following commands up until the next matching NDO for those lines in the range m through n which do not contain mtext in column c through d.

NDO End DO

Terminate a DO sequence. DO's and NDO's must be matched.

TERMINATION COMMANDS

END End

Updates the source file, saving the working file with all switch settings and file contents, and terminates the editor.

QUIT Quit

Terminates the editor, discarding the working file without making any changes to the source file.