Book 4

Editing the Source Program

DECtape Editor
(Editor)

Line Editor for Disk
(LINED)

Text Editor and Corrector
(TECO)

DECTAPE EDITOR (EDITOR)

Editor creates, adds to, or deletes from sequentially numbered source files recorded in lines of ASCII characters on a DECtape. Editor edits the source file; (the input and output files are the same). Fresh source files have editing space in each physical DECtape block. If the user has more edits for a block than will fit in it, an extra block in the DECtape is used and appropriately linked to the preceding and following logical blocks of the file. Editor provides a simple method of creating or modifying Macro or FORTRAN IV source programs.

Requirements

Minimum Core:	1K
Additional Core:	Not used
Equipment:	One DECtape unit for the reel containing the file(s) to be modified

Initialization

.R EDITOR)	Loads the DECtape Editor program.
*	Editor is ready to receive a command.

Commands

Initialize a File For Processing

Command	Function
Sn tA)	Select DECtape n and zero the directory.
Sn,filename.ext tA $	Select DECtape n, zero the directory, and create a file called filename.ext.

Command	Function
Sn, filename.ext)	Select DECtape n and locate filename.ext for processing.
Sn, filename.ext $	Select DECtape n and add a new file called filename.ext.

NOTE

All the above commands place Editor in the Command mode; i.e., the next typein is assumed to be one of the commands given below.

Insert a Line

Innnnn)	Insert the following typed line at line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or as a point (.). A point refers to the last line typed. If the line number already exists in the file, the line is replaced.
nnnnn aaaa.....a)	
nnnxx $)	
*	

Insert Multiple Lines

Innnnn, increment)	Insert the following typed lines, beginning at line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or as a point (.). Each time a line is entered, nnnnn is increased by the specified increment, and the result becomes the line number for the next insertion. Type $ after last line insertion.
nnnnn aaaa...aaa)	
nnnxx bbbb...bbb)	
nnnxx $)	
*	

Delete a Line

Dnnnnn)	Delete line number nnnnn from the currently open file; nnnnn can be specified as a line sequence number or as a point (.).
*	

Command	Function
Delete a Series of Lines	
Dmmmmm,nnnnn)	Delete lines mmmmm through nnnnn from the currently open file.
*	
Print a Line	
Pnnnnn)	Print line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or as a point (.).
nnnnn aaa...aaa)	
*	
Print a Series of Lines	
Pmmmmm,nnnnn)	Print lines mmmmm through nnnnn of the currently open file.
mmmmm aaa...aaa)	
.	
.	
nnnnn bbb...bbb)	
*	
Close the Current File	
E) (End of file)	Closes the currently open file. Another file can be opened on the same or a different DECtape via an Sn command, or a return can be made to Monitor to terminate Editor.
*	

Examples

.R EDITOR)	
*S1,VECTOR $	Select DECtape 1 and create a new file on DECtape 1 called VECTOR.
*I20,20)	Begin inserting at line sequence number 20, and increment this number by 20 each time a line is inserted. Switch to Text Mode.

```
                                    Editor responds with first line sequence number.
00020 DEFINE VMAG (A,B)↵            Operator types line of coding to be inserted,
00040 <MOVE 0,A↵                    followed by a carriage return.
00060 FMP 0↵
00080 MOVE 1,A+1↵
00100 FMP 1,1↵
00120 FAD 1↵
00140 MOVE 1,A+2↵
00160 FMP 1,1↵
00180 FAD 1↵
00200 JSR FSQRT↵
00220 MOVEM B↵
00240 $ $↵                          Typing $ terminates insertions and returns Editor
                                    to command mode.
*I20↵                               Change line number 00020.

00020 DEFINE VMAG (A,B,C)↵

*ILR*↵                              *ILR* indicates that the indexing increment has
                                    resulted in the next line number being equal to
                                    the line number of an already existing line (00040).
                                    Note that the indexing increment remains as 20
                                    until explicitly changed.

*I90↵                               Insert a line between lines 00080 and 00100.

00090 MOVE 1,C↵

*ILS*↵                              *ILS* indicates that the indexing increment has
                                    resulted in an existing line (00100) being skipped,
                                    since the next line addressed would be 00110.

*D180↵                              Delete line 00180.

*P20,220↵                           Print lines 00020 through 00220.

00020 DEFINE VMAG (A,B,C)↵
00040 <MOVE 0,A↵
00060 FMP 0↵
00080 MOVE 1,A+1↵
00090 MOVE 1,C↵
00100 FMP 1,1↵
00120 FAD 1↵
00140 MOVE 1,A+2↵
00160 FMP 1,1↵
00200 JSR FSQRT↵
00220 MOVEM B↵

*E↵                                 Close the currently open file.

*↑C↵                                Return to the Monitor.

.
```

Editor Diagnostic Messages

Message	Meaning
?DDE*	Device Data Error due to a write error or WRITE LOCK switch. Editor must be restarted.
?DEC*	DECtape directory is full.
?FAU*	Filename Already Used. A filename assigned to a new file already exists on the DECtape.
?ILC*	Illegal Command.
ILR *ILS*	Insert Line Replacement, Insert Line Skip. The line sequence increment specified for the insert function will cause the next existing line to be either replaced (R) or skipped (S). This is a warning message only and does not necessarily indicate an error.
?NCF*	Not a Current File.
?NFO*	No File Open. A command requiring an active file has been given but no file is currently open.
?NLN*	Nonexistent Line Number. A print (P) or delete (D) command refers to a nonexistent line.
?UNA*	Unit Not Available. The DECtape specified in an Sn command is assigned to another job.

LINE EDITOR FOR DISK (LINED)

LINED is similar to DECtape Editor, but operates on disk files instead of DECtape files. LINED is called in by typing

```
.R LINED
*
```

LINED responds with an asterisk to indicate it is ready to accept a command string.

Commands

LINED commands are very similar to those of Editor; consequently, only a brief summary is given here.

Command	Function
S filename.ext)	Select an existing file for editing.
S filename.ext $	Select (create) a new file for editing.
Innnnn)	Insert a line at line number nnnnn.
Innnnn,increment)	Insert multiple lines, starting at line number nnnnn and incrementing the line number each time.
Dnnnnn)	Delete the line at line number nnnnn.
Dmmmmm,nnnnn)	Delete lines mmmmm through nnnnn.
Pnnnnn)	Print line number nnnnn on the Teletype.
Pmmmmm,nnnnn)	Print lines mmmmm through nnnnn on the Teletype.
E)	End (close) the current file.
$	If in Insertion Mode, ignore current text line and return to LINED command level. If in Command Mode, print the next line.

NOTE

Files are written with standard protection (055). All blocks are assumed to have integral number of lines. Use the /A switch (line blocking) with PIP to put each file on disk.

Diagnostic Messages

The diagnostic messages for LINED are similar to those for Editor. The diagnostic message ?DEC* is not included, and the following message is added:

Message	Meaning
?CCL*	CCL error. Error occurred while referencing CCL command file.

Monitor Commands

To call in LINED and open a new file for creation:

Monitor Command	Equivalent CUSP Commands
.CREATE filename.ext)	.R LINED) *S filename.ext $

To call in LINED and open an existing file for editing:

Monitor Command	Equivalent CUSP Command
.EDIT filename.ext)	.R LINED) *S filename.ext

TEXT EDITOR AND CORRECTOR (TECO)*

TECO edits files recorded in ASCII characters on any standard device. It can perform simple editing functions as well as highly sophisticated search, match, and substitute operations, and operate upon arbitrary length character strings under control of commands which are themselves character strings (and contains the mechanisms necessary to exploit this recursiveness).

Requirements

Minimum Core:	4K
Additional Core:	Takes advantage of any additional core available. Each 1K additional core augments the basic 6,200+ - character buffer by 5K additional characters.
Equipment	One input device and one output device.

NOTE

TECO automatically requests more core to expand its buffer under any of the following conditions:

*PDP-10 TECO is based on PDP-1 TECO, developed at MIT by Daniel Murphy, and PDP-6 TECO, developed at MIT's project MAC by Stewart Nelson and Richard Greenblatt.

a. An insert by way of the "I" command or "X" (Q Register) will overflow the present memory boundaries.

b. The command acceptance routine needs more core.

c. The total number of characters in the Data Buffer falls below 3000, and an input command from a peripheral device (other than the user console) is executed. Thus, TECO maintains a Data Buffer of at least 3000 characters.

If TECO is successful at obtaining more core, the following message will be typed:

```
*10000 <IJS >$$
[5K CORE]                    nk Core, where n is
*                            new core size of job
```

If TECO is unsuccessful at obtaining the core request, the following message is typed:

```
STORAGE CAPACITY EXCEEDED
?
*
```

Initialization

```
.R TECO)        Loads the Text Editor and Corrector program.
*               TECO is ready to accept a command.
```

Basic Commands

NOTE

When typing command strings to TECO, the following points should be noted.

$ — One $ is used to terminate the text within a command string, where applicable; two successive $s terminate the entire command string sequence and generate a RETURN, LINE-FEED.

(RUBOUT) — The RUBOUT key can be used to erase the preceding typed-in character(s) of a command string. Each character erased is echoed back on the teletype (e.g., ABD (RUBOUT) DC...). Successive RUBOUTs can be used to erase more than one character.

N.B. To erase a carriage return (which generates RETURN, LINE-FEED), two RUBOUT's are required, one RUBOUT to erase the LINE-FEED and one to erase the RETURN.

Two successive tGs ((BELL) s) can be used to wipe out the entire command string currently being typed.

TECO commands in the form tx (where x is any character) can be entered by either holding down the CTRL key while striking the x key or typing up-arrow (shift N) followed by the x character. These alternatives are not true where tx is a character within a text string (such as in a Search argument); in this case, the CTRL key must be used.

A carriage return, line feed, ( ) is ignored in a TECO command string as long as it does not appear within a particular command, such as Insert. Examples of this are given on the following pages.

Select The Input Device

Command	Function
ERdev:filename.ext [proj,prog] $	Edit Read. Selects the input device and file (if specified).

dev:*

DTAn:	(DECtape)
PTR:	(paper tape reader)
DSK:	(disk)
MTAn:	(magnetic tape)
CDR:	(card reader)

filename.ext (DSK: or DTAn: only)

[proj,prog] (DSK: only)

NOTE

Device TTY: cannot be used here. See I (Insert) command.

Specified only if file is located in other than user's area.

EBdev:filename.ext $	Edit Backup. Selects an input and file to be edited (the input device, which will also be the output device for the edited file, must be the disk*). EB is intended to be used to keep a backup of a file during a debugging session, without the user having to invent a new name for each version of the file.

*If dev: is not specified, DSK: is assumed.

Command	Function

For example, the command string sequence

```
EBPROG1.MAC  $
.
.
editing
.
.
EF  $$
```

results in:

a. Reading from file PROG1.MAC on disk

b. Creating a new output file, which is initially given a temporary name of TECOnn.TMP, where nn is the user's job number in octal; incorporating the job number in the filename solves the problem of identifying temporary files belonging to multiple 100,100 users

c. Performing a number of RENAMEs following the EF command so that the input file becomes PROG1.BAK, any previous PROG1.BAK file is deleted, and the new output file becomes PROG1.MAC.

An ER command can be given following an EB command and before the EF command, but an intervening EW command is illegal and results in the error message ?22 (see Table 2-5). Even though an ER command may be given, the name of the final output file is still taken from the EB command.

Select The Output Device

EWdev:filename.ext [proj,prog] $	Edit Write. Selects the output device and file (if specified).
EZdev.filename.ext [proj,prog] $	Edit Zero. Selects the output device and file (if specified), and rewinds the tape (if magnetic tape) or zeros the directory (if DECtape).

dev:*	DTAn:	(DECtape)
	DSK:	(disk)
	MTAn:	(magnetic tape)
	PTP:	(paper tape punch)
	LPT:	(line printer)

*If dev: is not specified, DSK: is assumed.

Command	Function
	filename.ext (DSK: or DTAn: only) [proj,prog] (DSK: only) Specified only if file is located in other than user's area.
Terminate Output; Close File	
EF	End File. Terminate output on the current output file and close the file without selecting a new output file.
EX	Exit. The EX command finishes the current edit operation by writing out all remaining pages of a file, performing an EF (End of File) command, and then exiting to the Monitor. Thus, EX is equivalent to the command string 1000000PEF (BELL)
EG	Exit and Go. The EG command executes an EX command followed by the last Monitor command (COMPILE, LOAD, EXECUTE, or DEBUG) typed by the user. (See Chapter 9.)
Magnetic Tape Positioning	
EM	Edit Magtape. Rewind the currently selected input magnetic tape.
nEM	Depending upon the value of n, perform one of the following operations on the currently selected input magnetic tape.

n	Operation
1	Rewind tape to load point.
3	Write end of file.
6	Skip one record.
7	Backspace one record.
8	Skip to logical end of tape.
9	Rewind and unload tape.
11	Erase 3 in. of tape.
14	Advance tape one file.
15	Backspace tape one file.

NOTE

Throughout TECO, all numbers in command strings are interpreted as decimal.

Input Commands

Command	Function
Y	Yank. Read from current input device into buffer until a. A FORM character is read (i.e., a page has been input), or b. The buffer is more than 2/3 full and one of the following is encountered (1) Line Feed (2) Form Feed c. or a point 128 characters from the end of the buffer is reached. NOTE The FORM character, if read, does not enter the buffer. Any data previously residing in the buffer is destroyed. The pointer is positioned immediately before the first character in the buffer. Representative buffer size for 5K TECO: Total buffer capacity = approx. 11,200 characters 2/3 buffer capacity = approx. 7,460 characters 1 line-printer page = 7,200+ characters (120 char./line) (60 lines) 7,800+ characters (130 char./line)
A	Append. Read from the current input device and append the incoming data to information already residing in the buffer. Terminate reading on the same conditions as in Y. NOTE No previous data is destroyed. The pointer is not moved.

Output Commands

PW	Punch, Wait. Output the entire buffer to the selected output device, with a FORM character appended as the last character. Do not alter the contents of the buffer or move the pointer.

Command	Function
P	Equivalent to a PW command followed by a Y command (i.e., output the current contents of the buffer followed by a FORM character, and then read in more data from the input device).
np	Perform the P command n times.
m,nP	Output the m+1 through the nth character from the buffer to the current output file. Do not append a FORM character at the end. Do not alter the contents of the buffer or move the pointer.

Editing Commands

Move The Pointer

nj	Jump. Move pointer to the right of the nth buffer character and give the pointer symbol (.) the value of n. If n is omitted, set pointer in front of the first buffer character (same as 0J).
nC	Continue. Set the pointer to the right of the nth character beyond the pointer's present position (equal to .+nJ). If n is omitted, 1 is assumed.
nR	Reverse. Set the pointer to the left of the nth character prior to the pointer's present position (equal to .-n). If n is omitted, 1 is assumed.
nL	Lines. +n - Move the pointer to the right, stopping after it has passed over n LINE-FEED characters. -n - Move the pointer to the left, stopping after it has passed over n+1 LINE-FEED characters, then move to the right of the last LINE-FEED character passed over. If n is omitted, assume 1L.

Delete Text

nD	Delete. Delete n characters. +n - Delete n characters just to the right of the pointer. -n - Delete n characters just to the left of the pointer. If n is omitted, 1 is assumed.
nK	

Command	Function
nK	Kill. +n - Move the pointer to the right, stopping after it has passed over n LINE-FEED characters. Delete all characters the pointer passes over. -n - Move the pointer to the left, stopping after it has passed over n+1 LINE-FEED characters, then move it to the right of the last LINE-FEED character passed over. Delete all characters between this point and the pointer's previous position. If n is omitted, 1 is assumed.
m,nK	Delete the m+1 through the nth characters of the buffer. Set the pointer where the deletion occurred.

Insert Text

Itext... $	Insert. Insert the text following the "I" up to, but not including, the $ character, beginning at the current pointer position. Move the pointer to the right of the inserted material.
nI	Insert at the pointer location a character with an ASCII code of n (n must be a decimal value). Move the pointer to the right of the inserted character.
n\	Insert at the current pointer location the ASCII text representation of the decimal value of the expression n. Move the pointer to the right of the inserted text.
→| text... $	Insert at the current pointer location a (→|) character and the following text up to, but not including, the $ character. Move the pointer to the right of the inserted text.
@I/text/	Insert at the current pointer location the text which follows. The text is delimited by a character, /, which can be any character not appearing in the text.

Type Text

NOTE

T commands do not move the pointer.

Command	Function
nT	Type. Type out the string of characters beginning at the current pointer position and terminating after the nth LINE-FEED character is encountered. +n - Typeout n lines to the right of the current pointer position. -n - Typeout n lines to the left of the current pointer position. If n is omitted, the value is assumed to be 1.
m,nT	Type out the m + 1 through the nth characters of the buffer.

Stand-Alone Examples (Elementary)

Open an Input File

ERDTA5:SOURCE.MAC $	Open the input file called SOURCE.MAC located on DTA5.
ERDSK:SRCE.MAC[12,24] $	Open the input file called SRCE.MAC located in area 12,24 on the disk.
ERPTR: $	Open an input file on the paper tape reader.

Open an Output File

EWDTA3:EDITED.MAC $	Open an output file on DTA3 and call it EDITED.MAC.
EZDTA1:DEBUG.MAC $	Zero the directory on DTA1, open an output file on it, and call the file DEBUG.MAC.

Read a Page

Y	Read a page into the buffer from the current input file, destroying the previous contents of the buffer.
A	Read a page into the buffer, appending the data to the end of the information currently in the buffer.

Output Data

Command	Function
PW	Output the entire buffer, followed by a FORM character.
6P	Execute the WRITE and READ cycle six times.
12,50P	Write out the 13th through the 50th characters of the buffer.

Pointer Positioning

Y18J	Read in a page of information and position the pointer after the 18th character of the buffer.
5R	Then, move the pointer left to between characters 13 and 14.

Delete Text

J19C3D or	Move the pointer to the right of the 19th character in the buffer and then delete the next three characters to the right (characters 20, 21, and 22).
19,22K	Delete the 19+1 (20th) through the 22nd characters of the buffer.

Insert Text

J2LITAG: MOVE 1, AMT $	Move the pointer to a position following the second line of the buffer; insert the text TAG: MOVE 1, AMT between the second and third lines of the buffer.
69\	Insert the digits 69 in ASCII at the current pointer position (same as I69 or $ or 541571).
	NOTE Unless a \key is present, \ is typed with a SHIFT L.
→| ERROR IN JOB $	Insert a tab followed by the text ERROR IN JOB at the current pointer position.

Command	Function
@I#ERDSK:PROG1#)	Insert the text ERDSK:PROG $ at the current pointer position.

NOTE

The use of delimiters is the only method for inserting a $ in the text.

Typing Text

3T	Type out the first three lines of the buffer.
25,100T	Type out the 25+1 (26th) through the 100th character of the buffer.

Examples (Basic)

*R TECO)	
*ERDTA1:SCFILE.MAC1$$)	Open the file called SCFILE.MAC on DTA1 for input.
*EWDTA2:EDFILE.MAC1$$)	Open an output file on DTA2 and call it EDFILE.MAC.
*Y0,20T1$)	Read a buffer of information from the input file and type the first 20 characters of the buffer.
aaaaa.....aaaaa)	
*3LT1$)	Move the pointer to the right, stopping when three LINE-FEED characters have been encountered; type the text of the fourth line in the buffer.
bbbbb.....bbbbb)	
*ITHIS IS A SAMPLE INSERT) 1$)	Insert the text THIS IS A SAMPLE INSERT) between the third and fourth lines of the buffer, and position the pointer after the inserted material.
*10PT1$)	Write out the current buffer to the output device; read in and write out the next nine pages of data; read in the 11th page of data and position the pointer at the beginning of the buffer; type out the first line of the buffer.
ccccc.....ccccc)	
*K1$$)	Delete this line from the file; position the pointer at the beginning of the (now) first line in the buffer.

Command	Function
*EX$$)	Writes out all remaining pages of the file, performs an EF (End of File) command, and exits to the Monitor.
EXIT) ↑C)	Kill the job, deassign all devices, release core.

Advanced Commands

Search Commands

Summary

S text $ (Search) - Searches for text in current buffer only.

N text $ (Nonstop Search) - Searches for text through successive buffers by repeatedly writing out current buffer and reading in next buffer (similar to P command, but a form character is not inserted after outputting each buffer).

+text $ - Searches for text through successive buffers by repeatedly reading in new bufferful of information (Y command).

NOTES

All searches begin at the current location of the pointer.

Each search command can be preceded by the modifier characters (: and/or @).

: causes the search command to have a numeric value at completion;

0 if the search has failed (the requested text was not found) or -1 if the search was successful (the requested text was found).

@ indicates that the text to be matched is delimited by some character (same as in the @I command).

A numeric argument can appear following the modifiers (if any) but must precede the command. If the numeric argument is n, TECO searches for the nth occurrence of the text. If n is not used, the value of n is assumed to be 1.

If search is successful, the pointer is positioned to the right of the matched text. If the search fails, the pointer is positioned at the beginning of the buffer.

Use of special characters within text:

tS - Match any separator character (any character not a letter, number, period, dollar sign, or percent symbol).

tX - Match any (arbitrary) character. Used when the contents of some position within the text is unimportant.

tNx - Match any character except x.

tQ - Takes the next character literally, even if it is one of four special characters. For example, S tQ tX $ - Find the character tX.

See note under TECO, Basic Commands.

Search Commands Summary

Command	Action at End of Buffer	Action at End of File	Values Success	Values Fail	Typeout? if Failure
S	Failure	N/A	N/A	N/A	Yes
:S	Failure	N/A	-1	0	No
N	Performs a P command (but a FORM character is not inserted) and resumes search	Failure	N/A	N/A	Yes
:N	Performs a P command (but a FORM character is not inserted) and resumes search	Failure	-1	0	No
←	Performs a Y command (read only) and resumes search	Failure	N/A	N/A	Yes
:←	Performs a Y command (read only) and resumes search.	Failure	-1	0	No

Q-Register Commands

Q registers are provided for storing quantities, command strings, or buffer contents for later use. Thirty-six Q registers, labeled 0 through 9 and A through Z, are available.

Command	Function
nUi	Use. Places the numeric value n in Q-register i.
Qi	Q-register. Represents the current value in Q-register i
%i	Adds 1 to the value in Q-register i and represents the new value.
m,nXi	Xfer. Copies characters m+1 through the nth character of the buffer into Q-register i. Does not alter buffer contents or pointer.
nXi	Copies the buffer characters between the current pointer position and the nth LINE-FEED character in Q-register i
Gi	Get. Inserts the text contained in Q-register i into the buffer beginning at the current pointer location. Set the pointer to the right of the insertion.
[i	Pushes the contents of Q-register i onto the Q-register pushdown list.
]i	Pops the top entry of the Q-register pushdown list into Q-register i. The Q-register pushdown list is cleared each time two successive $s are typed.

Macro, Iteration, and Conditional Commands

Mi	Macro. Perform the text in Q-register i as a series of commands.
< >	Iteration brackets. When > is encountered, command interpretation is sent back to <.
n < >	Perform the commands within the iteration brackets n times.
;	If not in an iteration, an error results. If most recent search failed, send command interpretation to just beyond the matching >on the right; otherwise, no effect.
n;	If not in an iteration, an error results. If the value of n is 0 or greater, send command interpretation just past the matching >to the right; otherwise, no effect.
.' tag '.	Tag definition. Tag is the name of the location in which it appears in a command string.
O tag $	Go to the named tag, which must appear in the current macro or command string.
n"G	If n is Greater than or equal to 0, perform commands up to next '. Otherwise, skip to next '.

Command	Function
n"L	If n is Less than or equal to 0, perform commands up to next '. Otherwise, skip to next '.
n"N	If n is Not equal to 0, perform commands up to next '. Otherwise, skip to next '.
n"E	If n is Equal to 0, perform commands up to next '. Otherwise, skip to next '.
n"C	If n is a symbol Constituent (a letter, number, period, dollar sign, or percent symbol), perform commands up to next '. Otherwise, skip to next '.

NOTE

The double quotation mark (") and the single quotation mark (') symbols are matched in the same way as the left parenthesis symbol, (, and right parenthesis symbol,).

Numeric Values and Arguments in Command Strings

Many command string formats permit arguments with numeric values. The following characters may appear in a command string to develop these values in any instance where a numeric value is permissable.

0 through 9	Represent their corresponding numeric values.
B	Beginning. Equivalent to 0.
Z	Equivalent to the number of characters in the buffer.
.	Equivalent to the number of characters to the left of the current pointer position (or in other words, equal to the current pointer position).
Qi	Q-register. Equivalent to most recent numeric value placed in Q-register i.
nA	ASCII. Equivalent to ASCII value of character to right of pointer; n is used to differentiate this argument from an Append command (A) and has no other significance.
tH	Equivalent to value of elapsed time in 60ths of a second since midnight.
tF	Equivalent to the value of the console data switches.

Command	Function
↑E	Has the value of the form feed switch. If, during the last Y or A command execution, data transmission was terminated by a form feed character, ↑E has a value of -1, otherwise, the value is 0.
↑↑	(On Teletype Models 33 and 35, hold down both the CTRL and SHIFT keys and type N.) Equivalent to the ASCII value of the next character in the command string; this character is not interpreted as a command.
↑T	Typed Character. Stops command execution until user types a character on the Teletype; ↑T then becomes equivalent to the ASCII value of the character typed.
\	Equivalent to the value represented by the digits (or minus sign) immediately following the current pointer position. The value is terminated by the first nonnumeric character encountered. The pointer is positioned immediately following the value.
m+n m-n	Add Subtract } Take one or two arguments. A space is equal to +.
m*n m/n	Multiply Divide (truncates) } Take one or two arguments.
m&n	Logical AND; bitwise AND of binary representations m and n.
m#n	Logical IOR; bitwise inclusive OR of binary representations m and n.
( )	Operators +, -, *, /, #, and $ are normally performed left to right. This sequence can be overruled by use of parentheses.

NOTE

TECO does not assume that multiplication and division are always performed before addition and subtraction. Thus, to obtain the equivalent of a + (b * c), one must use the parentheses; otherwise, (a + b) * c is assumed.

n=	Causes the value of n to be typed out.
H	Abbreviation for B, Z. (0 through the last location of the buffer; in other words, the whole buffer).

NOTE

If a command takes two numeric arguments, a comma is used to separate them.

TECO Termination Commands

Command	Function
tC	Returns control to the Monitor without waiting for any I/O operations to finish.
tG (BELL)	Returns control to the Monitor after completing all current output requests.

Stand-Alone Examples (Advanced)

Search

J3SMOVE $ IM $	Within the current buffer, search for the third occurrence (3S) of the text MOVE, position the pointer immediately after it, and insert an M at that point.

Search for a Special Character

S↑NA $	Search for any character except A within the current buffer.
S↑S $	Search for any separator character within the current buffer.

Q-Registers, Macros, Iterations, and Conditionals

JØUN <S↓ $;%N>QN =	Count the number of LINE-FEED characters in the buffer as follows: 1. Position the pointer at the beginning of the buffer (J), 2. Place 0 in Q-register N(0UN), 3. Perform a search for a LINE-FEED character (S LINE-FEED $); if one is found, add 1 to Q-register N (;%N). Go back (< >) and repeat this cycle until the end of the buffer is reached and the test fails (;); at this point type out the contents of Q-register N(QN=).
J<SJUMPA $; -4DIRST $ >	Whenever JUMPA appears in the current buffer replace it with JRST.

Command	Function
	1. Position the pointer at the beginning of the buffer (J). 2. Search for JUMPA; when found, backspace the pointer four positions and delete the four characters passed over (;-4D). 3. Replace these four characters with the characters RST (IRST). 4. Repeat this routine (< >) until the test fails (end of the buffer has been reached) and exit (;) to >.

Placing a Command in a Q-Register for Later Execution

@I# JØUN <S↓ $)	1. Insert the text "JØUN <S)↓ $; %N >QN=" into the buffer (@I#.....#)
;%↑>QN = #HXP	2. Copy the contents of the buffer into Q-register P (HXP).
To Execute the Command:	
ERDTA3:FN.EX $ YMP	1. Read in a page of a file to search. (ERDTA3:FN.EX $ Y) 2. Execute the command stored in Q-register P (MP).

Reading in Text to be Inserted in Several Places in a File and Storing it in a Q-Register

ERPTR: $ YHXP)	1. Assume that the text to be inserted is on paper tape. Open an input file on the paper tape reader (ERPTR:); read the text into the buffer (Y); copy the contents of the buffer into Q-register P (HXP).
ERDTA4: TXTEDT $)	2. Open the input file to be edited and the output file to contain the edited version.
EWDSK: TXTEDU $) YNCALC: $ GP)	3. Read a page from the input file and initiate a search for the text CALC: . When found, insert the text stored in Q-register P at that point (GP).
NTOT: $ GP	4. Search for the text TOT: and, when found, insert the text stored in Q-register P after it.

Examples (Advanced)

Command	Function
.R TECO) *ERMTA1:$EM14EM$$)	Select MTA1 for input; rewind the tape (EM) and advance the tape one file (14EM).
EZDTA1:REVFIL$$)	Select DTA1 for output; zero the directory; open a file and call it REVFIL.
*YNTAXRT$ØLT) 1X1$$) aaaa...TAXRT aaaa.....aaaaa	Read in the first page from the input file; search for the text TAXRT; if it cannot be found, write the buffer out, read in the next page, search again, etc.; continue this cycle until either TAXRT is found or end of file is reached. If TAXRT is found, position the pointer at the beginning of the line containing it, type the line, and place the line in Q-register 1.
*JNTXRTE$ØLT, ØLT) G1$$) bbb...TXRTE bbb.....bbbbb	Search the buffer for the text TXRTE; if not found, write out the buffer, read the next page, search again; continue this cycle until either TXRTE is found or end of file is reached. If TXRTE is found, position the pointer at the beginning of the line containing it, type the line, and insert the contents of Q-register 1 immediately before that line.
*NTXTEND:$) J<SA$;1A-47"G1A-58"L-DIB$">) PWEF$$)	Read pages from the input file and write them on the output file until end of file (marked by the text TXTEND;) is found. At that point, move the pointer to the beginning of the buffer (J), and search for all As in the buffer (SA); if the character following the A is a digit, 0 through 9 (ASCII codes 48_{10} through 57_{10}), change the A to a B (1B); continue searching and modifying until end of buffer is reached; write out the last page and write end of file on the output device.
*↑G$$) EXIT) ↑C)	Return control to the Monitor after all output requests have been completed.

TECO Diagnostic Messages

Message	Meaning
?n	n is a decimal number associated with one of the list of error messages given in Table 2-5. TECO ignores the remainder of the command string and returns to the idle state. At this point, the user can type back ?, causing TECO to type out the command string terminated by the bad command.

Error List for ?n Messages

n_{10}	Meaning
1	TECO attempted to read commands beyond the terminating $$. This error is probably due to an unterminated @I or @S command, or to an unsatisfied O command.
2	Same as 1.
3	An attempt was made to supply more than two arguments to a command, either by the use of two commas or by "H,".
4	Too many right parentheses.
5	= command with no argument.
6	U command with no argument.
7	Q,U,X, or G command specifies an illegal Q-register (i.e., other than A through Z or 0 through 9).
8	In an X command, the second argument is not greater than the first.
9	In a G command, the Q-register does not contain text.
10	In a G command, the data in the Q-register is not in correct form (this is an internal error).
11	In an Ec command (e.g., ER, EW, EF, etc.), c is illegal.
12	File not found on LOOKUP.
13	Blank filename specified for directory device.
14	Project-programmer number specified does not have UFD.

Error List for ?n Messages

n_{10}	Meaning
15	Protection failure on disk.
16	File cannot be accessed because it is currently being written.
17	LOOKUP or ENTER returned error type 6 (not defined).
18	LOOKUP or ENTER returned error type 7 (no device).
19	Directory full on ENTER.
20	Requested I/O device not available.
21	Not assigned.
22	EW command between an EB command and its EF.
23	EM command given, but no input file open.
24	nEM command, where n is not in the range 1 to 16.
25	Internal error: EF after EB, but no input file is open.
26	Illegal character in filename.
27	Illegal character in project-programmer number.
28	Attempt to read an input page when no file has been opened for input.
29	I/O error on input device.
30	Attempt to output a page when no file has been opened for output.
31	Two arguments were supplied for an L command.
32	Attempt to move pointer beyond page.
33	A 2-argument command has its second argument less than the first argument.
34	Attempt to search for too long a character string.
35	Search command did not find the required string.
36	In an M command, the Q-register does not contain text.
37	In an M command, the data in the Q-register is not in correct form (this is an internal error).
38	Unmatched right angle bracket.

Error List for ?n Messages

n_{10}	Meaning
39	; encountered when not in iteration.
40	" command with no numeric argument, or "x where x is not G, L, E, N, or C.
41	This is the number typed out at the end of the ? command's dump of the command string in error. Refer to the number of the previous error.
42	A character has been encountered as an undefined command.
43	A tD command, when DDT has not been loaded with TECO.
44	Not enough core available from the Monitor.
45	A RENAME attempted with either a blank name or one already in use. Presumably due to a fault in the EB command.

Debugging Aids - As an aid in debugging macros and iterations, TECO can be set in the trace mode by typing ? as any character other than the first in a command string. When in Trace Mode, TECO types out each command as it is interpreted, interspersed with requested output. Typing a second ? in the same manner takes TECO out of Trace Mode; the ? can be typed each time it is desired to change the current mode.

The user can also type comments on his teletype sheet as he executes TECO by typing:

t Atext tA

This causes all text entered to be printed on the teletype (with the exception of terminating tA character).

NOTE

Since the terminator tA is not a command, it must be typed by holding the CTRL key down while typing A. It cannot be entered as "up arrow, A."

If DDT has been loaded along with TECO by the Linking Loader, control can be transferred to DDT by using the command tD.

Monitor Commands

To call in TECO and open a new file for creation:

Monitor Commands	Equivalent CUSP Commands
.MAKE filename.ext	.R TECO
* (text input commands) $$	*EWDSK:filename.ext $$
*EX $$	*(text input commands) $$
	*EX $$

To call in TECO and open an already existing file for creation:

Monitor Commands	Equivalent CUSP Commands
.TECO filename.ext	.R TECO
*(editing) $$	*EBDSK:filename.ext $ Y $$
*EX $$	*(editing) $$
	*EX $$