DEC-10-UTECA-A-D

decsystem10

INTRODUCTION TO TECO (TEXT EDITOR AND CORRECTOR)

This document represents the software as of version 23 of TECO.

digital equipment corporation · maynard, massachusetts

1st Printing May 1972

CONTENTS

CHAPTER 1	INTRODUCTION TO TECO	191
1.1	General Operating Procedure	191
1.2	Initialization	192
1.3	Special Symbols Used in this Document	193
1.4	General Command String Syntax	194
1.5	Erasing Commands	195
1.6	Command Arguments	196
CHAPTER 2	TECO COMMANDS	
2.1	Input Commands	197
2.2	Buffer Pointer Positioning	198
2.3	Text Typeout	200
2.4	Deletion Commands	201
2.5	Insertion Command	202
2.6	Output Commands	204
2.7	Special Exit Commands	205
2.8	Search Commands	206
CHAPTER 3	ERROR MESSAGES	

TABLES

1-1	Special Symbols	193
3-1	TECO Error Messages	210

CHAPTER 1
INTRODUCTION TO TECO

TECO, a very powerful text editor, enables the advanced DECsystem-10 user to edit any ASCII text with a minimum of effort. All editing can be accomplished by using only a few simple commands; or the user may select any of a large set of sophisticated commands such as character string searching, command repetition, conditional commands, programmed editing, and text block movement. In this description of TECO only the basic commands are described. If the user requires information about the more advanced uses of TECO, he can refer to the TECO manual in the DECsystem-10 Users Handbook.

TECO is a character-oriented editor. One or more characters in a line can be modified without retyping the rest of the line. Any sort of document can be edited: programs written in FORTRAN, COBOL, MACRO-10, or any other language; memoranda; specifications; and other types of arbitrarily formatted text. TECO does not require that line numbers or any other extraneous information be associated with the text.

1.1 GENERAL OPERATING PROCEDURE

TECO operates on ASCII data files. A file is an ordered set of data on some peripheral device. In the case of TECO, a data file is some type of document. An input file may be a named file on disk or DECtape, a file on magnetic tape, a deck of punched cards, or a punched paper tape. An output file can be written onto any of these same devices. The input file for a given editing operation is the file to which the user wishes to make changes. If the user is using TECO to create a new file, there is no input file. The output file is either the newly created file or the edited version of the input file. An output file is not required if the user wishes merely to examine a file without making any changes.

In general, the process of editing proceeds as follows. The user first specified the file he wishes to edit and then reads in a "page" of text. A page is normally an amount of text that is intended for a single sheet of paper. Form feeds are used to separate a document into pages. On input, TECO interprets form feeds as end-of-page indicators. It is not required, however, that a document be so divided into pages. If a form feed is not encountered, TECO simply reads as much text as will reasonably fit into its editing buffer. For the purposes of this document, the word page is used to mean the segment of text in TECO's editing buffer.

When a page has been read into the buffer, the user can modify this text by using the various editing commands. When he has finished editing the page, he outputs it and reads in the next page. This process continues until, after the last page has been output, the user closes the output file. If there are several pages where no editing is required, there are commands which may be used to skim over them.

1.2 INITIALIZATION

The two main uses of TECO are (1) to create a new disk file, and (2) to edit an existing disk file. These are the only uses of TECO described in this document. In particular, the use of TECO with devices other than disk is not described. The beginner can get around this limitation by using PIP to transfer files to and from disk. (Refer to the PIP manual in the DECsystem-10 Users Handbook for information about PIP.)

The two main uses of TECO are so common that there are direct monitor commands to initialize TECO for executing them. The command

.MAKE filename.ext ↵

is used to initialize TECO for creating a new disk file. Filename.ext is the name that the user gives to the new file. The filename can be from one to six alphanumeric characters. This is followed (optionally) by a period (.) and a filename extension of from one to three alphanumeric characters. The most commonly used filename extensions are:

.F4	for FORTRAN source programs
.CBL	for COBOL source programs
.MAC	for MACRO-10 source programs

The MAKE command opens a new disk file to receive output from TECO and gives it the name specified by the user. Once the file has been opened it is then actually created by using the insert and output commands, which are explained in sections 2.5 and 2.6 of this document.

The command

.TECO filename.ext ↵

is used to initialize TECO for editing an existing disk file, named filename.ext. The filename and filename extension must be exactly the same as those of the file that is to be edited. The TECO command opens the specified file for input by TECO and opens a new file, with a temporary name, for output of the edited version. When output of the new version is completed, the original version of the file is automatically renamed filename.BAK, and the newly edited version is given the name of the original file. The filename extension .BAK is used for backup files.

After TECO has been initialized for a particular job, it responds by typing an asterisk (*). The asterisk indicates that TECO is ready to accept commands; it is typed at the beginning of TECO's operation and at the completion of execution of every command string.

Examples:

```
.MAKE EARNNG.F4↵
*
```

This command initializes TECO for creation of a new disk file called EARNNG.F4. The extension .F4 is used because the file is to be a FORTRAN source file.

```
.TECO  LIB40.MAC↵
*
```

This command initializes TECO for editing the existing disk file LIB40.MAC. At the completion of editing, TECO automatically changes the name of the original version of LIB40.MAC to LIB40.BAK and gives the name LIB40.MAC to the new version.

NOTE

The TECO command cannot be used to edit a file which has the filename extension .BAK. To edit a backup file the user must first rename the backup file. For example, to edit the file LIB40.BAK the user should proceed as follows:

```
.RENAME  LIB40.OLD=LIB40.BAK↵
.TECO  LIB40.OLD↵
*
```

1.3 SPECIAL SYMBOLS USED IN THIS DOCUMENT

Table 1-1
Special Symbols

Symbol	Character Represented	Comment
↵	Carriage Return	Whenever the RETURN key is typed, TECO automatically appends a line feed to the carriage return.
(S)	Altmode	On most terminals, the altmode key is labeled "ALTMODE", but on some it is labeled "ESC" or "PREFIX". Since the altmode is a non-printing character, TECO indicates that it has received an altmode type-in by echoing a dollar sign ($).
↑C	Control C	This character is typed by typing the letter C while holding down the CTRL key. Other control characters are represented in similar fashion.

Table 1-1 (Cont)
Special Symbols

Symbol	Character Represented	Comment
(FORM)	Form Feed	Form feed is typed by typing (↑F) (control F).
↓	Line Feed	This symbol is used only when a line feed is explicitly typed. It is not used for the line feed which is automatically assumed when a carriage return is typed.
→|	Tab	Tab is typed by typing (↑I) (control I).
Δ	Space	This symbol is used occasionally for emphasis.
(RO)	Rubout	This key is used to nullify a character erroneously typed in a command string. Its use is explained fully in Section 1.5.

1.4 GENERAL COMMAND STRING SYNTAX

TECO commands are usually given by typing the one- or two- letter name of the command. However, many of the commands take arguments. Some typical examples are shown below, to give the reader an idea how TECO commands look. These commands are fully explained later in the manual.

```
L
PW
ISAMPLE($)
3K
```

TECO commands may be given one at a time. However, it is usually more convenient to type, in a single command string, several commands that form a logical group. An example of a command string is shown below.

```
*IHEADING($)NTAG:($)2LT($)($)
```

A command string may be typed after TECO indicates its readiness by printing an asterisk. Command strings are formed by merely writing one command after another. Command strings are terminated by typing two consecutive altmodes.

Execution of the command string begins only after the double altmode has been typed. At that point each command in the string is executed in turn, starting at the left. When all commands in the string have been executed, TECO prints another asterisk, indicating its readiness to accept another command.

If some command in the string cannot be executed because of a command error, execution of the command string stops at that point, and an error message is printed. Commands preceding the bad command are executed. The bad command and those following it are not executed.

1.5 ERASING COMMANDS

Typographical errors, if discovered while typing a command string, may be "erased" by use of the rubout key. This process is best explained by an example.

*3LKILEIF ERICXON

After typing this much of the command string, the user discovers that he has misspelled the name "Ericson." To nullify his error, he types three successive rubouts. As he does this, TECO responds by retyping the characters which are being rubbed out.

*3LKILEIF ERICXON (RO) N (RO) O (RO) X

Of course, rubout is a non-printing character so the actual line looks like this:

* 3LKILEIF ERICXONNOX

Once he has rubbed out the bad character, the user continues the command string from the last correct character.

* 3LKILEIF ERICXONNOXSON ($) 0LT ($)($)

The actual function of the rubout character is to delete the last typed character in the command string. Consequently, if the bad character is not the last in the string, all characters back to that point must be deleted. Rubout characters do not enter the command string.

An entire command string may be erased, if it has not yet been terminated, by typing two successive ↑G (control G) characters.

Example:

* 3LKILIEF ERICXON ↑G ↑G	↑G ↑G causes the entire command string to be rejected. TECO types a new asterisk and awaits a new command.

1.6 COMMAND ARGUMENTS

There are two types of arguments for TECO commands. Some commands require numeric arguments and some require alphanumeric (text) arguments.

Numeric arguments, and also all numeric type-outs by TECO, are decimal integers. Numeric arguments always precede the command to which they apply. A typical example of a command taking a numeric argument is the command to delete three characters: "3D".

Alphanumeric arguments are textual arguments meant to be interpreted as ASCII code by TECO. Alphanumeric arguments always follow the command to which they apply, and they must always be terminated by an altmode. Examples of alphanumeric arguments are (1) text to be inserted, and (2) character strings to be searched for.

Example:

*ISOMETHING $$ The argument is "SOMETHING".

As shown in the above example, the altmode used to terminate an alphanumeric argument may also serve as one of the two altmodes necessary to terminate a command string. Any ASCII character except null, altmode, and rubout may be included in an alphanumeric argument.

CHAPTER 2
TECO COMMANDS

2.1 INPUT COMMANDS

The Y (yank) command first clears the editing buffer and then reads the next page of the input file into the buffer.

A single Y command is automatically performed by the command

.TECO filename.ext)

so that when editing with this command the first page of the input file is automatically read in before TECO prints the first asterisk.

The Y command may be used to delete entire pages of a file, since the editing buffer is completely cleared before the input is performed.

The A (append) command reads in the next page of the input file without clearing the current contents of the editing buffer. This command is used to combine several pages of a document. When the A command is used, the form feed separating the page already in the buffer and the page to be read in is removed. Thus after the A command the two pages are combined into one.

If the editing buffer does not have enough room to accommodate an A command which has been given, TECO automatically expands its buffer and then executes the A command. The user is notified of this action by a message of the following form

[3K CORE]

If sufficient core is not available to allow buffer expansion, the user is notified by an error message.

NOTE

On either an A or a Y command the form feed terminating the page to be read in is not actually read into the buffer. It is removed on input and a single form feed is appended to the end of the buffer when the buffer is output.

Examples:

.TECO REPORT.CBL↵ * —	This command, as part of the process of initializing TECO for editing the disk file REPORT.CBL, automatically clears the buffer and then reads in the first page of the file.
*Y$$ * —	This command deletes the entire contents of the buffer and then reads in the next page of the input file.
*AA$$ * —	Read the next two pages of the input file into the buffer, combining them with the page already in the buffer.
*A$$ [4K CORE] * —	The buffer is expanded as required by the A command. In most cases this message need be of no concern to the user. It is important only if the system is low on core and does not have swapping capability.

2.2 BUFFER POINTER POSITIONING

Since TECO is a character-oriented editor, it is very important that the user understand the concept of the buffer pointer. The position of the buffer pointer determines the effect of many of the editing commands. For example, insertion and deletion always take place at the current position of the buffer pointer.

The buffer pointer is simply a movable position indicator. It is always positioned between two characters in the editing buffer, or before the first character in the buffer, or after the last character in the buffer. It is never positioned "on" a particular character, but rather before or after the character. The pointer may be moved forward or backward over any number of characters.

The J command moves the buffer pointer to the beginning of the buffer, i.e., to the position immediately before the first character in the buffer.

The ZJ command moves the pointer to the end of the buffer, i.e., to the position following the last character in the buffer.

The C command advances the pointer over one character in the buffer. The C command may be preceded by a (decimal) numeric argument. The command nC moves the pointer forward over n characters. (The pointer cannot be advanced beyond the end of the buffer.)

The R command moves the pointer backward over one character in the buffer. This command may also be preceded by a numeric argument. The command nR moves the pointer backward over n characters. (The pointer cannot be moved backward beyond the beginning of the buffer.)

The L command is used to advance the buffer pointer or move it backward, on a line-by-line basis. The L command takes a numeric argument, which may be positive, negative, or zero, and is understood to be one (1) if omitted.

The action of the L command with various arguments is best explained in a more concrete way. Suppose the buffer pointer is positioned at the beginning of line b, or at some position within line b.

The command L, or 1L, advances the pointer to the beginning of line b+1, i.e., to the position following the line feed which terminates line b.

The command nL, where n >0, advances the pointer to the beginning of line b+n.

The command 0L moves the pointer to the beginning of line b. If the pointer is already at the beginning of line b, nothing happens.

The command -L moves the pointer back to the beginning of line b-1.

The command -nL moves the pointer back to the beginning of line b-n.

NOTE

After execution of a Y command, the buffer pointer is always positioned before the first character in the buffer. (The Y command automatically executes an implicit J command.) The A command does not change the position of the buffer pointer.

In examples, the position of the buffer pointer is often represented in this manual by the symbol ↑ just below the line of text.

Examples:

*J3L$$ * *ZJ-2L$$ * *L4C$$ * *0L2R$$ *	The J command moves the pointer to the beginning of the first line in the buffer. The 3L command then moves it to the beginning of the fourth line. This moves the pointer to the beginning of the next to last line in the buffer. Advances the pointer to the position following the fourth character in the next line. 0L moves the pointer back to the beginning of the line it is currently on. Then 2R moves it back over the carriage return-line feed pair which terminates the preceding line.
ABCDEF ↑	In this example of text stored in the buffer, the position of the buffer pointer is shown to be between B and C.

2.3 TEXT TYPE-OUT

Various parts of the text in the buffer can be typed out for examination. This is done by use of the T command. Just what is typed out depends on the position of the buffer pointer and the argument given. The T command never moves the buffer pointer.

The command T types out everything from the buffer pointer through the next line feed. Thus, if the pointer is at the beginning of a line, the command T causes that line to be typed out. If the pointer is in the middle of a line, T causes the portion of the line following the pointer to be typed.

The command nT (n>0) is used to type out n lines, i.e., everything from the buffer pointer through the nth line feed following it.

The command 0T types out everything from the beginning of the current line up to the buffer pointer. This is useful for determining the position of the pointer.

The command HT types out the entire contents of the buffer.

The user, especially one new to TECO, should use the T command often, to make sure the buffer pointer is where he thinks it is.

During execution of any T command, the user may stop the terminal output by typing the ↑O (control O) character. This command causes TECO to finish execution of the command string, omitting all further type-out. The ↑O command does not carry over to the next command string.

Examples:

```
*0LT($)($)
ENTIRE LINE TYPED
*
```

This command string is used to move the pointer back to the beginning of a line and then type out the entire line. It is frequently used after insertion and search commands.

```
*0TT($)($)
ENTIRE LINE TYPED
*
```

This command string causes the entire line to be typed without moving the pointer. It is useful after insertion and search commands when it is not convenient to move the pointer back to the beginning of the line.

```
*2T($)($)
EF
GHIJKL
*
```

If the buffer contains the text below with the pointer between D and E,

ABCD↑EF ↵ ↓
GHIJKL ↵ ↓
MNOPQR ↵ ↓

this command causes the typeout shown.

"ABCD" is not typed because these characters precede the pointer. MNOPQR is not typed because these characters follow the second line feed.

2.4 DELETION COMMANDS

Characters are deleted individually by using the D command. The command D deletes the character immediately following the buffer pointer. The command nD, where n >0, deletes the n characters immediately following the pointer. The commands -D and -nD delete the character or the n characters, respectively, which immediately precede the buffer pointer.

Lines are deleted using the K command. The K command may be preceded by a numeric argument, which is understood to be 1, if omitted. The command nK (n > 0) deletes everything from the current position of the pointer through the nth line-feed character following the pointer. The command HK deletes the entire contents of the buffer.

At the conclusion of a D or K command the buffer pointer is positioned between the characters which precede and follow the deletion.

Examples:

The editing buffer contains the following three lines of text, and the pointer is positioned between the G and H.

ABCDEFG↑HIJKLM ↵↓
NOPQRSTUVWXYZ ↵↓
1234567890 ↵↓

*4D $$	Delete HIJK.
*	
*-D $$	Delete G.
*	
*-3D $$	Delete EFG.
*	
*7D $$	Delete HIJKLM ↵↓ but do not delete the line feed at the end of the first line.
*	
*K $$	Delete HIJKLM ↵↓.
*	Since the carriage return and line feed at the end of the first line are deleted, the text in the buffer after this command would be: ABCDEFGNOPQRSTUVWXYZ ↵↓ 1234567890 ↵↓
*2K10D $$	This would leave the buffer containing only ABCDEFG ↵↓
*	

```
*0LK $$
*
```
This is the command string that is required to kill (delete) the entire first line.

```
*L2K $$
*
```
This kills the last two lines.

```
*HK $$
*
```
Kill the entire buffer.

2.5 INSERTION COMMAND

The only insertion command is the I command. The ASCII text that is to be inserted into the buffer is typed immediately after the letter I. The text to be inserted is terminated by an altmode.

Any ASCII character except null, altmode, and rubout may be included in the text to be inserted. Specifically, spaces, tabs, carriage returns, form feeds, line feeds, and control characters are all allowed. If a carriage return is typed in an insertion, it is automatically followed by a line feed.

The text to be inserted is placed in the buffer at the position of the buffer pointer, i.e., between the characters. At the conclusion of the insertion command the buffer pointer is positioned at the end of the insertion.

Any number of lines may be inserted with a single I command. For the user's protection, however, no more than 10 to 20 lines should be inserted with each I command.

Examples:

If the buffer contains ABCD↑EF) ↓ with the pointer between D and E, the command

```
*IXYZ $$
*
```
produces ABCDXYZ↑EF) ↓

```
*I)
$$
*
```
produces ABCD) ↓
↑EF) ↓

```
*I↓
$$
*
```
produces ABCD↓
↑EF) ↓

```
*3RI_Δ $4CI_Δ $$
*
```
produces A_ΔBCDE_Δ↑F) ↓

```
*I (FORM)
$$
*
```
This command is used to separate the page in the buffer into two pages. Both pages, however, remain in the buffer. They are not actually separated until output.

```
*JILINE ONE)
LINE TWO )
LINE THREE)
($)($)
*
```

This example shows insertion of several lines of text at the beginning of the buffer.

```
*KI)
($)($)
*
```

This is the command string used to delete the tail of a line without removing the carriage return-line feed at the end of the line. If the buffer contains

AB↓CD)↓
EFGH)↓

This command will produce

AB)↓
↑EFGH)↓

2.6 OUTPUT COMMANDS

The command P causes (1) the entire contents of the editing buffer to be output to the output file and (2) an implicit Y command to be performed which reads in the next page of the input file. This command is used after editing of a given page is complete and the user is ready to move on to the next page.

The P command may be used with a positive numeric argument to skim over several pages. Specifically, the nP command causes the n consecutive pages of the input file, starting with the page in the editing buffer, to be output, and then the n+1st page to be yanked in.

The PW command merely outputs the page currently in the editing buffer. It does not clear the buffer, it does not read in any more text, and it does not move the buffer pointer. This command is used when creating a new file. It is also used to output the last page of a file.

If the buffer is empty, the PW and P commands have no effect.

The EF command must be used to close the output file after all output to it is complete. EF is normally used after the PW command which outputs the last page of the file.

Examples:

```
*PWEF ($)($)
*
```

This is the command string usually used to close out a file when the last page of the file is in the buffer.

```
*PT ($)($)
FIRST LINE
*
```

This command string outputs the current page, reads in the next page, and then types the first line of the new page.

```
*8P $$
*
```

If, for example, page 6 of a document is in the editing buffer, this command causes pages 6 through 13 of the document to be output, one after the other, and then reads in page 14.

2.7 SPECIAL EXIT COMMANDS

The EX command is used to conclude an editing job with a minimum of effort. Its use is best shown by an example.

Suppose the user is editing a 30-page file and suppose that the last actual change to the file is made on page 10. At this point the user gives the command

```
*EX $$
.
```

In this case the action performed by TECO is equivalent to the command string 20PPWEF, with an automatic return to the monitor at the end. Thus, the action of TECO is (1) to rapidly move all the rest of the input file on to the output file, (2) close the output file, and (3) to return control to the monitor.

The EG command is even more efficient. This command performs exactly the same functions as the EX command, but after that it causes re-execution of the last COMPILE, LOAD, EXECUTE, or DEBUG command attempted before TECO was called.

For example, suppose the user gives the command

```
.COMPILE PLOT.F4↵
```

To request compilation of a FORTRAN source program, but the compiler discovers errors in the code. The user would then call TECO to correct these errors:

```
.TECO PLOT.F4↵
*
```

When all the errors are edited, the user exits from TECO with the command

```
*EG $$
```

This causes the COMPILE command to be executed again on the file PLOT.F4, after TECO has finished output of the file.

Any TECO job may be aborted by using the standard return-to-monitor command: ↑C ↑C (control C typed twice). However, if this command is typed before the output file is closed, the output file is lost.

If no input or output operations are in progress a single ↑C is sufficient to exit from TECO to the monitor. In such a case, the user may reenter TECO without destroying the job he was previously executing. This is illustrated in the following example.

```
.TECO    SOURCE.MAC)          A TECO job is started.
*ICOMMENTS ($)($)
*↑C                           The user exits to perform a few simple monitor
                              commands.
.DEASSIGN LPT)
.DAYTIME)
24-MAY-72   10:34
.REE)                         The user reenters TECO. The previous buffer
*                             is still intact.
```

2.8 SEARCH COMMANDS

In many cases the simplest way to position the buffer pointer is by using a character string search. A search command causes TECO to scan through the text until a specified string of characters is found, and then to position the pointer at the end of this string. There are two main search commands.

The S command is used to search for a character string within the editing buffer. The string to be searched for is specified as an alphanumerical argument following the S command. This argument must be terminated by an altmode. The character string to be searched for may contain any ASCII character except null, altmode, or rubout.

The S command may be preceded by a numerical argument n > 1. This argument is used to search for the nth occurrence of a character string. Thus a 2S command searches for the second occurrence of the particular character string, skipping the first occurrence. If n is omitted, n = 1 is assumed.

Execution of the S command begins at the position of the buffer pointer and continues to the end of the buffer. If the specified character string is not found in this range, an error message is printed and the buffer pointer is set to the beginning of the buffer.

Examples:

```
*SA →|B ($)($)     This causes the pointer to be positioned after the
*                  B in the first occurrence of the string
                   A - tab - B past the current position of the pointer.
```

```
*J2SNAME $$
*
```

This causes the pointer to be positioned after the second occurrence of the string "NAME" in the buffer.

```
*S20↓
TAG: $ 0LT $$
TAG:     REST OF LINE
*
```

This moves the pointer to the position just following the colon in the string "20↓ TAG:", then repositions the pointer to the beginning of the line (just before the "TAG:") and types out the entire line starting with "TAG:".

Warning: When attempting a search it is very easy to overlook an occurrence of the search string preceding the one which the user desires. For example, he may want to move the pointer after the word "AND" but erroneously position it after a preceding occurrence of a word like "THOUSAND". For this reason the user, especially the novice, is strongly urged to execute a T command to ascertain the position of the pointer after each search command.

Example:

```
*SWORD $ 0TT $$
FORMAT(1X,'WORD')
*I␣WORD2 $$
*
```

Here the user wishes to insert "␣WORD2" after "WORD". He wisely types out the line to make sure he is at the right place, before inserting "WORD2".

The other principle search command is the N command. The difference is that an S search ends at the end of the current buffer, whereas an N search does not. An N search begins like an S search, but if the character string is not found in the current buffer, an automatic P command is executed. The current page is outputted, the next page read in, and the search continued on the new page. This process continues until either the string is found or the input file is exhausted.

If the N search does find the specified character string, the pointer is positioned at its end.

If the string is not found, an error message is generated. In this case the user caused himself a fair amount of delay. If an N search fails, the user must close the file with an EX command, then reopen it and try the N search again with a character string that can be found. The user is strongly urged to be careful when typing search character strings. Remember also that a search string must be terminated with an altmode.

Example:

```
*NSTRING - 3D $$
?SRH CANNOT FIND "STRING-3D"
*EX $$
.TECO filename.ext
*NSTRING $ - 3D $$
*
```

Here the user meant to search for the character string "STRING", and to delete the last three characters of the string. However, he forgot to terminate the search string with an altmode and this caused the unsatisfied search request error message (?SRH).

CHAPTER 3
ERROR MESSAGES

When TECO encounters an illegal command or a command that for any other reason cannot be executed, an error message is printed on the user's terminal. Such messages contain a three-character code of the form ?aaa and a one-line description of the error.

To get more information about the error, the user can type a slash (/) immediately after he receives the error message. TECO will type an additional message that describes the error in more detail. All three parts of the error messages from TECO are given in Table 3-1.

When an error message is generated, the command to which it refers is not executed, the remainder of the command string is ignored, and TECO retruns to the idle state by typing an asterisk and awaiting a new command string.

The novice user is especially warned that there are a great many TECO commands that have not been described in this introductory material. Almost every letter of the alphabet and many of the special characters have meanings as TECO commands. Hence, the user should be careful when typing command strings. The beginner should probably stick to relatively short command strings.

In the following table, all TECO error messages are listed, even though some of them refer to the more advanced commands not described in this manual. Error messages referring to the advanced commands will probably be encountered by the user of this introductory material only if he has typed an unintended command letter.

The complete set of TECO commands is fully described in the TECO manual in the DEC-system-10 Users Handbook. Since most editing can be done using only the basic commands covered in this introductory material, most users should be able to get along without the more advanced description for some time. The novice should gain complete mastery of the basic commands before attempting to use any of the advanced commands.

Table 3-1
TECO Error Messages

?ARG	Improper Arguments The following argument combinations are illegal: 1) , (no argument before comma) 2) m,n, (where m and n are numeric terms) 3) H, (because H=B,Z is already two arguments) 4) ,H (H following other arguments)
?BAK	Cannot Delete Old Backup File Failure in rename process at close of editing job initiated by an EB command or a TECO command. There exists an old backup file filnam.BAK with a protection<nnn> such that it cannot be deleted. Hence the input file filnam.ext cannot be renamed to "filnam.BAK". The output file is closed with the filenam "nnnTEC.TEMP", where nnn is the user's job number. The RENAME UUO error code is nn.
?COR	Storage Capacity Exceeded The current operation requires more memory storage than TECO now has and TECO is unable to obtain more core from the monitor. This message can occur as a result of any one of the following things: 1) command buffer overflow while a long command string is being typed, 2) Q-register buffer overflow caused by an X or [command, 3) editing buffer overflow caused by an insert command or a read command.
?COS	Contradictory Output Switches The GENLSN and SUPLSN switches may not both be used with the same output file.
?EBD	EB with Device dev Is Illegal The EB command and the TECO command may be specified only with file structured devices, i.e., disk and DECtape.
?EBF	EB with Illegal File filnam.ext The EB command and the TECO command may not be used with a file having the filename extension ".BAK" or with a file having the name "nnnTEC.TMP". Where nnn is the user's job number, the user must either use an ER-EW sequence, or rename the file.
?EBO	EB, EW, or EZ Before Current EB Job Closed After an output file has been opened by a TECO command or an EB command, no further EB, EW, or EZ commands may be given until the current output file is closed.
?EBP	EB Illegal Because of File filnam.ext Protection The file filnam.ext cannot be edited with an EB command or a TECO command because it has a protection <nnn> such that it cannot be renamed at close time.

Table 3-1 (Cont)
TECO Error Messages

?EEE	Unable to Read Error Message File An error, whose code was typed previous to this error message, has occurred, and while TECO was trying to find the proper error message in the error message file, one of the following errors occurred: 1) the error message file, TECO.ERR, could not be found on device SYS:, 2) an input error occurred while TECO was reading the file TECO.ERR, 3) The error message corresponding to that error code is missing from TECO.ERR, 4) the user's TECO job does not currently have enough room for a buffer to read the error message into, and no more core can be obtained from the monitor, 5) for some strange reason device SYS: could not be initialized for input.
?EMA	EM with Illegal Argument nn The argument n in an nEM command must be greater than zero.
?EMD	EM with No Input Device Open EM commands apply only to the input device, and so should be preceded by an ER (or equivalent) command. To position a tape for output, that unit should be temporarily opened for input while doing the EM commands.
?ENT-00	Illegal Output Filename "filnam.ext" ENTER UUO failure 0. The filename "filnam.ext" specified for the output file cannot be used. The format is invalid.
-01	Output UFD dev:[pj,pg] Not Found ENTER UUO failure 1. The file filnam.ext[pj,pg] specified for output by an EW, EZ, or MAKE command cannot be created because there is no user file directory with project-programmer number [pj,pg] on device dev.
-02	Output Protection Failure ENTER UUO failure 2. The file filnam.ext[pj,pg] specified for output by an EW, EZ, EB, MAKE, or TECO command cannot be created either because it already exists and is write-protected <nnn> against the user, or because the UFD it is to be entered into is write-protected against the user.
-03	Output File Being Modified ENTER UUO failure 3. The file filnam.ext specified for output by an EW, EZ, EB, MAKE, or TECO command cannot be created because it is currently being created or modified by another job.
-06	Output UFD or RIB Error ENTER UUO failure 6. The output file filnam.ext cannot be created because a bad directory block was encountered by the monitor while the ENTER was in progress. The user may try repeating the EW, EB, or TECO command, but if the error persists, it is impossible to proceed. Notify your system manager.
-14	No Room or Quota Exceeded on dev: ENTER UUO failure 14. The output file filnam.ext cannot be created because there is no more free space on device dev:, or because the user's quota is already exceeded there.

Table 3-1 (Cont)
TECO Error Messages

-15	Write Lock on dev: ENTER UUO failure 15. The output file filnam.ext cannot be created because the output file structure is write-locked.
-16	Monitor Table Space Exhausted ENTER UUO failure 16. The output file filnam.ext cannot be created because there is not enough table space left in the monitor to allow the ENTER. The user may try repeating the EW, EB, or TECO command, but if the error persists he will have to wait until conditions improve.
-23	Output SFD not Found ENTER UUO failure 23. The output file filnam.ext cannot be created because the sub-file-directory on which it should be ENTERed cannot be found.
-24	Search List Empty ENTER UUO failure 24. The output file filnam.ext cannot be created because the user's file structure search list is empty.
-25	Output SFD Nested too Deeply ENTER UUO failure 25. The output file filnam.ext cannot be created because the specified SFD path for the ENTER is nested too deeply.
-26	No Create for Specified SFD Path ENTER UUO failure 26. The output file filnam.ext cannot be created because the specified SFD path for the ENTER is set for no creation.
-nn	ENTER Failure nn on Output File filnam.ext The attempted ENTER of the output file filnam.ext has failed and the monitor has returned an error code of nn. This error is not expected to occur on an ENTER. Please send the TTY printout showing what you were doing to DEC with an SPR form.
?EOA	nEO Argument Too Large The argument n given with an EO command is larger than the standard (maximum) setting of EO=n for this version of TECO. This must be an older version of TECO than the user thinks he is using; the features corresponding to EO=n do not exist.
?FNF-00	Input File filnam.ext Not Found LOOKUP UUO failure 0. The file filnam.ext specified for input by an ER, EB, or TECO command was not found on the input device dev.
-01	Input UFD dev:[pj,pg] Not Found LOOKUP UUO failure 1. The file filnam.ext[pj,pg] specified for input by an ER, EB, or TECO command cannot be found because there is no User File Directory with project-programmer number [pj,pg] on device dev.
-02	Input Protection Failure LOOKUP UUO failure 2. The file filnam.ext[pj,pg] specified for input by an ER, EB, or TECO command cannot be read because it is read-protected <nnn> against the user.

Table 3-1 (Cont)
TECO Error Messages

-06	Input UFD or RIB Error LOOKUP UUO failure 6. The input file filnam.ext cannot be read because a bad directory block was encountered by the monitor while the LOOKUP was in progress. The user may try repeating the ER, EB, or TECO command, but if the error persists all is lost. Notify your system manager.
-16	Monitor Table Space Exhausted LOOKUP UUO failure 16. The input file filnam.ext cannot be read because there is not enough table space left in the monitor to allow the LOOKUP. The user may try repeating the ER, EB, or TECO command, but if the error persists he will have to wait until system conditions improve.
-23	Input SFD not Found LOOKUP UUO failure 23. The input file filnam.ext cannot be found because the sub-file-directory on which it should be looked up cannot be found.
-24	Search List Empty LOOKUP UUO failure 24. The input file filnam.ext cannot be found because the user's file structure search list is empty.
-25	Input SFD Nested too Deeply LOOKUP UUO failure 25. The input file filnam.ext cannot be found because the specified SFD path for the LOOKUP is nested too deeply.
-nn	LOOKUP Failure nn on Input File filnam.ext The attempted LOOKUP on the Input file filnam.ext has failed and the monitor has returned an error code of nn. This error is not expected to occur on a LOOKUP. Please send the TTY printout showing what you were doing to DEC with an SPR form.
?FUL	Device dev: Directory Full ENTER UUO failure n. The file filnam.ext specified for output by an EW or MAKE command cannot be created on DECtape dev because the tape directory is full.
?IAB	Incomplete <...> or (...) in Macro A macro contained in a Q-register and being executed by an M command contains an iteration that is not closed within the Q-register by a >, or a parenthetical expression that is not closed within the Q-register by a).
?ICE	Illegal Control-E Command in Search Argument A search argument contains a (↑E) command that is either not defined or incomplete.
?ICT	Illegal Control Command ↑<char> in text Argument In order to be entered as text in an Insert command or search command, all control characters (↑@ - ↑H and ↑N - ↑←) must be preceded by ↑R or ↑T. Otherwise they are interpreted as commands. The control character "↑<char>" is an undefined text argument control command.

Table 3-1 (Cont)
TECO Error Messages

?IDV	Input Device dev Not Available Initialization failure. Unable to initialize the device dev for input. Either the device is being used by someone else right now, or else it does not exist in the system.
?IEC	Illegal Character "<char>" After E The only commands starting with the letter E are EB, EF, EG, EH, EM, EO, ER, ET, EU, EW, and EZ. When used as a command (i.e., not in a text argument) E may not be followed by any character except one of these.
?IEM	Re-Init Failure on Device dev After EM Unable to re-initialize the device dev after executing an EM command on it. If this error persists after retrying to initialize the device with an ER command (or EW command if output to the device is desired), consult your system manager.
?IFC	Illegal Character "<char>" After F The only commands starting with the letter F are FS and FN. When used as a command (other than EF or in a text argument) F may not be followed by any character other than one of these.
?IFN	Illegal Character "<char>" in Filename File specifications must be of the form dev:filnam.ext[m,n] (\$) where dev, filnam, and ext are alphanumeric, and m and n are numeric. No characters other than the ones specified may appear between the EB, ER, EW, or EZ command and the altmode terminator ((\$)).
?ILL	Illegal Command <char> The character "<char>" is not defined as a valid TECO command.
?ILR	Cannot Lookup Input File filnam.ext to Rename It Failure in rename process at close of editing job initiated by an EB command or a TECO command. Unable to do a LOOKUP on the original input file dev:filnam.ext in order to rename it "filnam.BAK". The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The LOOKUP UUO error code is nn.
?INP-nn0000	Input Error nn0000 on File filnam.ext. A read error has occurred during input. The input file filnam.ext has been released. The user may try again to read the file, but if the error persists, the user will have to return to his backup file. The input device status word error flags are nn0000. (Note: This number represents the I/O status word (rh) with bits 22-35 masked out.) (040000 -- block too large). (100000 -- parity or checksum error). (140000 -- block too large and parity error). (200000 -- device error, data missed). (240000 -- block too large and device error). (300000 -- parity error and device error). (340000 -- block too large, parity error, and device error).

Table 3-1 (Cont)
TECO Error Messages

	(400000 -- improper mode). (440000 -- block too large and improper mode). (500000 -- parity error and improper mode). (540000 -- block too large, parity error, and improper mode). (600000 -- device error and improper mode). (640000 -- block too large, device error, and improper mode). (700000 -- parity error, device error, and improper mode). (740000 -- block too large, parity error, device error, and improper mode).
?IOS	Illegal Character "<char>" in I/O Switch The only valid characters in switches used with file selection commands are the alphabetic characters.
?IQC	Illegal command " <char> The only valid "commands are "G, "L, "N, "E, "C, "A, "D, "V, "W, "T, "F, "S, and "U.
?IQN	Illegal Q-register Name "<char>" The Q-register name specified by a Q, U, X, G, %, M, [,], or * command must be a letter (A through Z) or a digit (0 through 9).
?IRB	Cannot Rename Input File filnam.ext to filnam.BAK Failure in rename process at close of editing job initiated by an EB command or a TECO command. The attempt to rename the original input file filnam.ext to the backup filename "filnam.BAK" has failed. The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The RENAME UUO error code is nn.
?IRN	Cannot RE-Init Device dev for Rename Process Failure in rename process at close of editing job initiated by an EB command or a TECO command. Cannot reinitialize the original input device dev in order to rename the input file filnam.ext to "filnam.BAK". The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number.
?ISA	n Argument with Search Command The argument preceding a search command indicates the number of times a match must be found before the search is considered successful. This argument must be greater than 0.
?MAP	Missing ' In attempting to execute a conditional skip command (a " command whose argument does not satisfy the required condition) no ' command closing the conditional execution string can be found. Note: n"...' strings must be complete within a single macro level.

Table 3-1 (Cont)
TECO Error Messages

?MEE	Macro Ending with E A command macro being executed from a Q-register ends with the character "E". This is an incomplete command. E is the initial character of an entire set of commands. The other character of the command begun by E must be in the same macro with the E.
?MEF	Macro Ending with F A command macro being executed from a Q-register ends with the character "F" (not an EF). This is an incomplete command. F is the initial character of an entire set of commands. The other character of the command begun by F must be in the same macro with the F.
?MEO	Macro Ending with Unterminated O Command The last command in a command macro being executed from a Q-register is an O command with no altmode to mark the end of the tag-name argument. The argument for the O command must be complete within the Q-register.
?MEQ	Macro Ending with " A command macro being executed from a Q-register ends with the " character. This is an incomplete command. The " command must be followed by one of the characters G, L, N, E, C, A, D, V, W, T, F, S, or U to indicate the condition under which the following commands are to be executed. This character must be in the Q-register with the ".
?MEU	Macro Ending with ↑ A command macro being executed from a Q-register ends with the ↑ character. This is an incomplete command. The ↑command takes a single character text argument that must be in the Q-register with the ↑.
?MIQ	Macro Ending with <char> A command macro being executed from a Q-register ends with the character "<char>". This is an incomplete command. The <char> command takes a single character text argument to name the Q-register to which it applies. This argument must be in the same macro as the <char> command itself.
?MLA	Missing < There is a right angle bracket not matched by a left angle bracket somewhere to its left. (Note: an iteration in a macro stored in a Q-register must be complete within the Q-register.)
?MLP	Missing (Command string contains a right parenthesis that is not matched by a corresponding left parenthesis.
?MRA	Missing > In attempting to exit from an iteration field with a ; command (or to skip over an iteration field with a 0 argument) no > command closing the iteration can be found. Note: iteration fields must be complete within a single macro level.

Table 3-1 (Cont)
TECO Error Messages

?MRP	Missing) The command string contains, within an iteration field, a parenthetical expression that is not closed by a right parenthesis.
?MUU	Macro Ending with ↑↑ A command macro being executed from a Q-register ends with control-↑ or ↑↑. This is an incomplete command. The ↑↑ command takes a single character text argument that must be in the Q-register with the ↑↑.
?NAE	No Argument Before = The command n= or n== causes the value n to be typed. The = command must be preceded by either a specific numeric argument or a command that returns a numeric value.
?NAI	No Altmode after nI Unless the EO value has been set to 1, the numeric insert command nI must be immediately followed by altmode.
?NAQ	No Argument Before " The " command must be preceded by a single numeric argument on which the decision to execute the following commands or skip to the matching ' is based.
?NAU	No Argument Before U The command nUi stores the value n in Q-register i. The U command must be preceded by either a specific numeric argument or a command that returns a numeric value.
?NCS	No Command String Seen Prior to *i The *i command saves the preceding command string in Q-register i. In this case no command string has previously been given.
?NFI	No File for Input Before issuing an input command (Y or A) it is necessary to open an input file by use of an ER, EB, or TECO command.
?NFO	No File for Output Before giving an output command (PW, P, N, EX, or EG) it is necessary to open an output file by use of an EB, EW, EZ, MAKE, or TECO command.
?NTQ	No Text in Q-register x Q-register x, specified by a G or M command, does not contain text.
?OCT	"8" or "9" in Octal Digit String In a digit string preceded by ↑O, only the octal digits 0-7 may be used.
?ODV	Output Device dev Not Available Initialization failure. Unable to initialize the device dev for output. Either the device is being used by someone else right now, or it is write locked, or else it does not exist in the system.

Table 3-1 (Cont)
TECO Error Messages

?OLR	Cannot Lookup Output File dev:filnam.ext to Rename It Failure in rename process at close of editing job initiated by an EB command or a TECO command. The special LOOKUP on the output file filnam.ext required for DECtape in order to rename the file to 'filnam.ext" has failed. The original input file filnam.ext has been renamed "filnam.BAK", but the output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The LOOKUP UUO error code is nn.
?OUT-nn0000	Output Error nn0000 - Output File nnnTEC.TMP Closed An error on the output device is fatal. The output file is closed at the end of the last data that was successfully output. It has the filename "nnnTEC.TMP", where nnn is the user's job number. See Section 4.3 for a recovery technique. The output device status word error flags are nn0000. (Note: This number represents the I/O status word (rh) with bits 22-35 masked out.) (000000 -- end of tape). (040000 -- block number too large: device full or quota exceeded). (100000 -- parity or checksum error). (140000 -- block number too large and parity error). (200000 -- device error, data missed). (240000 -- block number too large and device error). (300000 -- parity error and device error). (340000 -- block number too large, parity error, and device error). (400000 -- improper mode or device write locked). (440000 -- block number too large and improper mode). (500000 -- parity error and improper mode). (540000 -- block number too large, parity error, and improper mode). (600000 -- device error and improper mode). (640000 -- block number too large, device error, and improper mode). (700000 -- parity error, device error, and improper mode). (740000 -- block number too large, parity error, device error, and improper mode).
?PAR	Confused Use of Parentheses A string of the form (...<...) has been encountered. Parentheses should be used only to enclose combinations of numeric arguments. An iteration may not be opened and not closed between a left and right parenthesis.
?POP	Attempt to Move Pointer Off Page with J, C, R, or D The argument specified with a J, C, R, or D command must point to a position within the current size of the buffer, i.e., between 0 and Z, inclusive.
?PPN	Illegal Character "<char>" in Project-programmer Number Project-programmer numbers in file specifications must be given in the form [m,n] where m and n are octal digit strings separated by a comma. No characters other than the ones specified may appear between the enclosing brackets.

Table 3-1 (Cont)
TECO Error Messages

?RNO	Cannot Rename Output File nnnTEC.TMP Failure in rename process at close of editing job initiated by an EB command or a TECO command. The attempt to rename the output file nnnTEC.TMP to the name "filnam.ext" originally specified in the EB or TECO command has failed. The original input file filnam.ext has been renamed "filnam.BAK", but the output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The RENAME UUO error code is nn.
?SAL	Second Argument Less Than First In a two-argument command, the first argument must be less than or equal to the second.
?SNA	Initial Search with No Argument A search command with null argument has been given, but there was no preceding search command from which the argument could be taken.
?SNI	; Not in an Iteration The semicolon command may be used only with a string of commands enclosed by angle brackets, i.e., in an iteration field.
?SRH	Cannot Find "<text>" A search command not preceded by a colon modifier and not within an iteration has failed to find the specified character string "<text>". After an S search fails the pointer is left positioned at the beginning of the buffer. After an N or ← search fails the last page of the input file has been input and, in the case of the N, output, and the buffer cleared. Note that when this message occurs, the text string printed includes all control-character commands included in the search argument.
?STC	Search String Too Long The maximum length of a search string is 80 characters including all string control commands and their arguments.
?STL	Search String too Long The maximum length of a search string is 36 character positions, not counting extra characters required to specify a single position.
?TAG	Missing Tag !xxx! The tag !xxx! specified by an O command cannot be found. This tag must be in the same macro level as the O command referencing it.
?TAL	Two Arguments with L The L command takes at most one numeric argument, namely, the number of lines over which the buffer pointer is to be moved.
?TTY	Illegal TTY I-O Device A terminal may be specified as an input-output device in an ER, EW, EZ, or MAKE command only if it is not being used to control an attached job, the user's own terminal included.

Table 3-1 (Cont)
TECO Error Messages

?UCA	Unterminated ↑A Command A ↑A message type-out command has been given, but there is no corresponding ↑A to mark the end of the message. ↑A commands must be complete within a single command level.
?UFS	Macro Ending with Unterminated File Selection Command The last command in a command macro being executed from a Q-register is a file selection command (ER, EW, EB, or EZ) with no altmode to mark the end of the file specifications. The file selection command must be complete within the Q-register.
?UIN	Unterminated Insert Command An insert command (possibly ar @ insert command) has been given without terminating the text argument at the same macro level.
?UIS	Undefined I/O Switch "/xxx" The switch "/xxx" is not defined with either input or output file selection commands. The only switches currently defined for input or output file selection commands are /GENLSN and /SUPLSN.
?USR	Unterminated Search Command A search command (possibly an @ search command) has been given without terminating the text argument at the same macro level.
?UTG	Unterminated Tag A command string tag has been indicated by a ! command, but there is no corresponding ! to mark the end of the tag. Tags must be complete within a single command level.
?UUO	Illegal UUO Internal error. The illegal instruction <lh,rh> has been encountered at address nnnnnn. This is caused by either a TECO bug or a monitor bug. Please give this printout to your system manager, or submit it to DEC with an SPR.