U.S. Department of the Interior

Geological Survey

USGS Mineralogy Laboratory User's Guide to the TECO Editing Program for the DEC RT-11 Operating System

(Part C of the USGS Mineralogy Laboratory User's Guide to the DEC RT-11 Operating System)

By

Richard E. Phillips and Phoebe L. Hauff

Open-File Report 82-177

1982

CONTENTS

INTRODUCTION TO THE TECO EDITING PROGRAM

The file editing program used with this computer system is known as TECO. TECO, an acronym for Text Editor and Corrector, is a character-oriented editor rather than a line editor. TECO allows the user to create or modify files consisting of ASCII characters, such as those containing data, source programs, or manuscripts.

In the TECO section of this manual, the word "text" is used to represent any group of ASCII characters in a file that is to be edited by TECO. In other words, "text" may consist of numbers, alphabetic characters, or any other members of the ASCII character set.

The TECO editor uses a portion of the computer's memory known as a "buffer" to hold part of the file's contents while that file is being edited or created. Text can be entered directly into the buffer by the user or it can be read into the buffer from an existing file. The buffer occupies a finite area of memory and can therefore contain only a limited amount of material at one time. This varies with the amount of space available in the computer's memory. Once editing is complete on the portion of the file held in the buffer, it is written into an output file and deleted from the buffer to make space available for additional input of text from the original file. This process is diagrammed in figure T-1.

This manual will give the user a working knowledge of TECO, some of its basic concepts, an outline and discussion of the more useful parts of the TECO command set, and examples showing TECO applied to some U.S. Geological Survey applications.

Much of the information contained in this manual is taken from the DEC "TECO Manual-Version 28," which serves as the prime reference for questions regarding TECO.

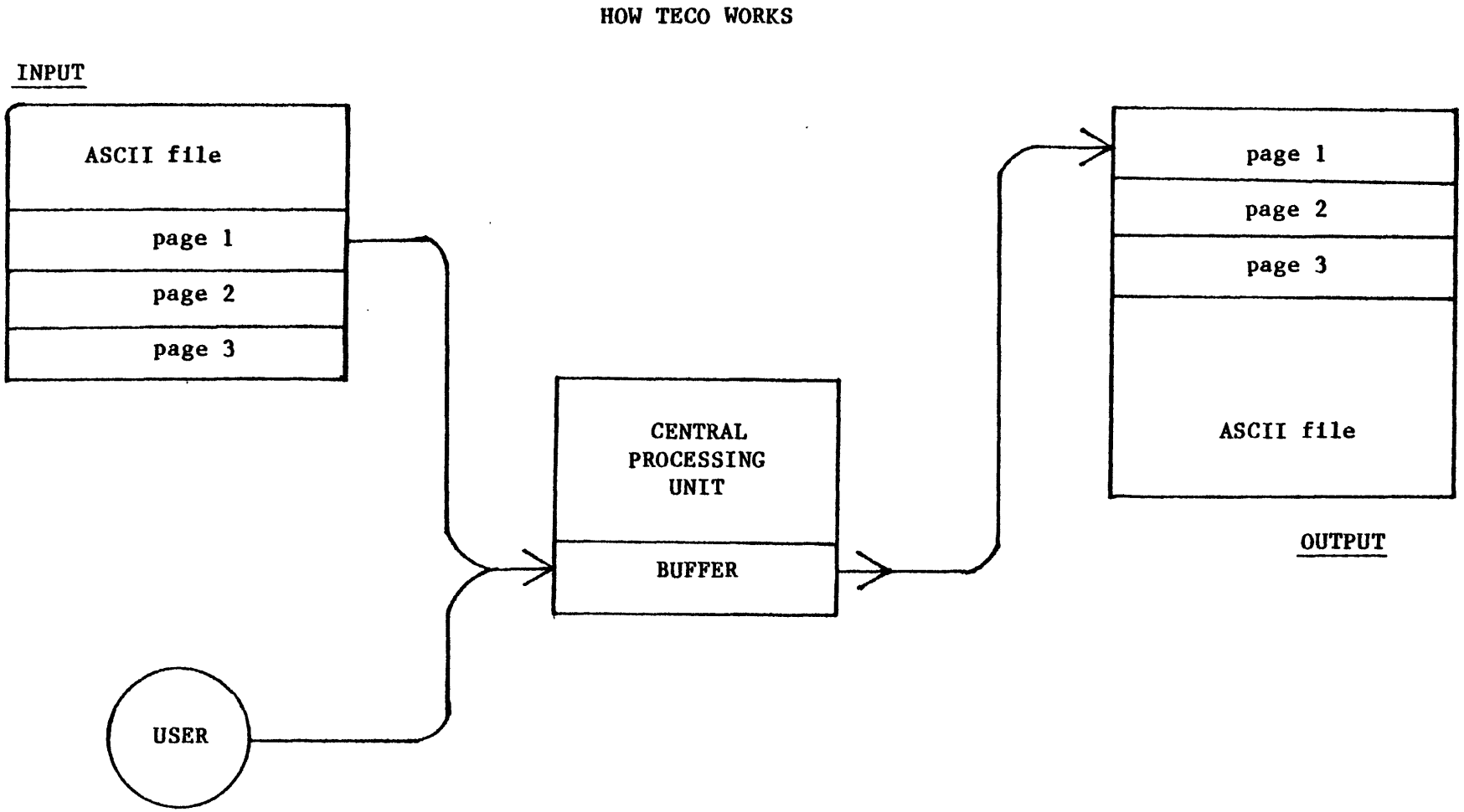

Figure T-1.--Diagrammatic representation of the processes of building a TECO file.

SOME BASIC TECO CONCEPTS

Before TECO can be utilized, there are some basic concepts of its structure and nomenclature that must be understood. The following section describes the more important and useful of these.

***** An "*" printed on the left-hand margin indicates that TECO is in memory and that the editor is ready to accept commands. This is similar to the period printed by the RT-11 monitor.

$ This is the symbol typed on the terminal when the ESC (ESCAPE) key is depressed; it does not represent a dollar sign. The ESC key is typed singularly to terminate certain commands and twice to execute a command or command string.

COMMANDS These generally consist of one- or two-character codes that direct the editor to manipulate text characters or lines of text. The codes are alphabetic characters and may be preceded by numeric characters which designate how many times the command is to be executed. Other commands are followed by ASCII characters that give additional information needed for the execution of the command.

COMMAND STRINGS Several commands may be strung together before they are all executed. Commands to which ASCII characters have been appended must be separated from following commands by a single "$" (ESC). The commands in the command string are executed from left to right.

INCORRECT COMMANDS

Whenever TECO detects an incorrect command, an error message is printed and command execution stops at that point. Immediately after the error message is printed, an "*" is printed indicating that TECO is ready to accept new commands. The commands following the error in the command string are ignored.

?

The faulty command in a command string can be located by typing a question mark (?). TECO will then print out all commands in that string up to and including the character which caused the error.

TYPING ERRORS

Typing errors are corrected in the same way as with the RT-11 operating system.

DELETE

The DELETE key is used for single character erasure. A "/" is not printed when the DELETE key is used, as is the case at the RT-11 monitor level. However, TECO reprints the deleted character.

CTRL/U

The CTRL/U command is used to delete the entire command line. This command is executed by typing a U while holding the CTRL (CONTROL) key down.

POINTER

The most important concept in TECO is that of the "pointer." The pointer indicates the current position of the editor in the buffer. This pointer does not point to a character; it points between two characters. There are commands to position the pointer at any point in a line, or at the beginning or end of the text in the buffer. Once positioned at the designated place, the pointer serves as a "target" for any editing commands. The user must understand how to manipulate the pointer to be able to successfully operate within TECO.

NON-PRINTING CODES

Line Feed
Carriage Return
Space

Included in the main body of textual material in a file are codes that are inserted whenever the LINE FEED, SPACE bar, or RETURN key is typed. These are all known as non-printing codes because a character is not printed on the terminal when these keys are typed. The code for the RETURN key actually consists of two codes; one for the carriage return followed by one for the line feed. When the completed file is printed on a terminal, no specific character is typed when one of these codes is encountered by the computer, but the action specified by the code is taken; that is, a space is inserted when the "space code" is read.

PAGE

TECO breaks up large units of text (contained in a file or entered by the user) into smaller units known as "pages." TECO edits one page of text at a time. Page manipulation commands are used to read a page into the buffer or to write a page from the buffer to the output file.

The amount of text contained in a page can be limited by inserting a special code at the end of each page. For example, if the user wants the page size to be 60 lines of text, the end-of page code is inserted after every 60 lines of text. This method is discussed on page 22 of the DEC "TECO Manual."

If the main body of text being edited does not contain end-of-page codes, TECO uses its own method to define the size of a page. In this case, TECO considers one page to consist of the amount of text that can be entered directly or read into the buffer until it is three-quarters full. Upon reaching this point, TECO will

terminate input to the buffer after the next line-feed code is encountered. This is the method normally used in the applications at the USGS Mineralogy Laboratory.

Under normal operating conditions, over 150 lines of text with 80 characters per line can be entered into the buffer at one time. Many more lines of text can be entered if each line contains less than 80 characters per line.

.EDIT

This command loads the TECO.SAV file into memory. However, the actual entry into TECO is more complicated and explained in detail later in this manual.

TO EXIT FROM TECO

There are several commands that allow the user to exit from TECO. The most commonly used is EX. This returns the user to the monitor command level of the RT-11 operating system. When this command is executed, the TECO program moves all the remaining text in the buffer or the input file to the output file and then closes the output file. All the edited text and the unedited portion of the original text is now saved in the output file.

EX

↵

This is the symbol used in this manual for a carriage return/line feed.

THE TECO COMMAND SET--ABBREVIATED

The TECO Command Set is large and gives the user a great amount of editing versatility. However, the average user needs to know only a small number of the commands to satisfy most of his file editing needs. In the following paragraphs, basic commands are introduced and information on their use is given. In addition, some of the common error and warning messages issued by TECO when commands are used incorrectly are also introduced.

The commands are grouped according to their functions. The group headings used here are the same as those used in the DEC "TECO Manual" so that additional commands in each group are easily accessible. The "TECO Pocket Guide" also lists these commands, but only gives a brief description of each one. Included in this manual are some commands for file selection, text buffer input/output, pointer positioning, text buffer type-out, text editing, and character string searching.

The error and warning messages are introduced along with the commands to which they refer.

FILE SPECIFICATION COMMANDS

If the material to be edited is already contained in a file, this file will be known as the input file to the text buffer. After being edited in the buffer, the material is written to another file known as the output file. Input and output files are selected either by TECO file specification commands or by using the RT-11 "EDIT" command and its options.

Once the user is operating within TECO, the input and output files may be changed with two-letter TECO file specification commands. The three most commonly used commands are:

ER (edit read)
EW (edit write)
EB (edit backup)

Each of these must be followed by the file name. Because the "EDIT" command is more commonly used for entering TECO and specifying input and output files, the above three commands will not be discussed in this manual. The reader is referred to the DEC "TECO Manual," section 2.1 for their descriptions.

THE RT-11 "EDIT" COMMAND

The "EDIT" command loads TECO into memory and also reads the first page of text into the buffer from the input file. The options for the "EDIT" command are:

/CREATE
/INSPECT
/OUTPUT

The "EDIT" command is followed by the file name and extension.

The general format for each of the "EDIT" command options along with a description of the option and specific examples follows:

EDIT/INSPECT device:file name.extension↵

This command opens the specified file as the input file for the text buffer. This option is used when the contents of a file needs to be inspected, but not changed.

Example:

EDIT/INSPECT MC1:QUARTZ.DAT ↵

The command in this example does three things: 1) Loads TECO into memory; 2) opens the file QUARTZ.DAT on the MC1: storage device for input to the buffer; and 3) reads the first page of the file into the buffer so the user may inspect its contents.

EDIT/CREATE device:file name.extension ↵

This command is used to create a new file. A file with the name and extension specified by the user is opened on the specified storage device to receive output from the text buffer. It is important to remember that once material has been placed in the output file it cannot be re-entered into the buffer unless the file is closed and then re-opened as an input file.

Example:

EDIT/CREATE MIN1.LST ↵

The command in this example loads TECO into memory and creates a new file, MIN1.LST, on the default storage device.

EDIT device:file name.extension ↵

When the "EDIT" command is used without any options, the specified file is opened for input to the text buffer and an output file with the same specifications is created.

This form of the command has a very useful safety feature; when this command is used the original input file is always retained. This can be very useful if for some reason TECO is aborted or the computer unexpectedly shuts down. In this situation the output file, which contains the edited text, will be lost, but the

original file will remain-with its original name and extension.

If the user exits from TECO in a normal manner, as with the "EX" command, the original file and the edited file are retained. However, the extension of the original file is changed to .BAK, which indicates that it is a

.BAK

"backup" file. The edited file is given the name and extension of the original file.

This is the command to use if the edited file is to have the same specifications (same name, extension and storage device) as the original file and if a copy of the original file, unedited, is to be retained.

Example:

EDIT RLØ:STARTS.COM ↵

The command in this example does four things: 1) loads TECO into memory; 2) designates the STARTS.COM file on the RLØ: storage device as the input file; 3) reads the first page of this file into the buffer; and 4) opens an output file with the same specifications as the input file.

Upon exiting TECO, the original file will be labeled STARTS.BAK and the new, edited version will be labeled STARTS.COM.

EDIT/OUTPUT:device:file name.extension device:file name.extension ↵
(output file) (input file)

This command allows the user to specify both an input and an output file. If the specifications for the input file are different from the specifications for the output file, the input file will remain unchanged upon exiting from TECO. At least one of the three file identifiers (device name, file name, or extension) must be different.

% SUPERSEDING EXISTING FILE

If both files have the exact same name and extension the warning "% SUPERSEDING EXISTING FILE" will be typed on the terminal. This indicates that when TECO is exited (as with the "EX" command), only one file will remain--the output file with edited text. No copy of the original file will remain; it will be deleted.

Example:

EDIT/OUTPUT:MC1:ZEOL1.DAT MC1:ZEOL.DAT ↵

The command in this example does four things: 1) loads TECO into memory; 2) designates ZEOL.DAT as

the input file; 3) designates ZEOL1.DAT as the output file; and 4) reads the first page of ZEOL.DAT into the buffer. The text in ZEOL.DAT is to be edited, and the edited text is to be written in ZEOL1.DAT. Since the two files have different names, the file ZEOL.DAT will be retained after exiting from TECO.

PAGE MANIPULATION COMMAND

The page manipulation command controls the transfer of textual material between the input/output files and the text buffer. It permits pages of text to be read into the buffer from an input file and written from the buffer into the output file. The PULL (P or nP) command is discussed below.

P (Pull)	Writes the contents of the text buffer into the output file, clears the buffer, and reads the next page from the input file into the buffer.
nP	Executes the P command "n" times, where n is an integer ≥ 1. A value of 1 is assumed if n is not specified.

BUFFER POINTER MANIPULATION COMMANDS

The buffer pointer is the only means of specifying the location within a block of text in which insertions, deletions, or corrections are to be made. The following commands permit the buffer pointer to be moved to a position between any two adjacent characters in the buffer. TECO positions the pointer before

the first character in the buffer after "EDIT" is executed or after every "P" command is executed. The JUMP (J, ZJ), LINE (L, nL), CHARACTER (C, nC) and REVERSE (R, nR) commands are introduced below.

J (Jump)	Moves the pointer to the beginning of the buffer, before the first character.
ZJ	Moves the pointer to a position immediately following the last character in the buffer.
L (Line)	Advances the pointer to the beginning of the next line.
nL	Executes the "L" command n times, where n is any integer. A positive value of n moves the pointer to the beginning of the nth line following the current pointer position. A negative
-nL	value moves the pointer backward n lines and positions it at the beginning of the nth line preceding the current
ØL	position. If n is zero, the pointer is moved to the beginning of the line on which it is currently positioned.
C (Character)	Moves the pointer forward across one character.

nC	Executes the "C" command n times. A positive value of n moves the pointer forward across
-nC	n characters. A negative value of n moves the pointer backward across n characters. If n is zero, the pointer position is not changed.
R (Reverse)	Moves the pointer backwards across one character.
nR	Executes the "R" command n times.

These commands may be used to move the buffer pointer within a page but not across page boundaries. If a "C" command attempts to move the pointer beyond the beginning or end of the buffer, the error message

?POP Pointer off page

is printed and the command is ignored.

If an "L" command attempts to exceed the page boundaries in this manner, the pointer is positioned at the boundary that would have been exceeded. The command "-1000L" would position the pointer before the first character in the buffer. The command "1000L" would position the pointer after the last character in the buffer. No error message is printed in either case.

BUFFER POINTER MANIPULATION AND NON-PRINTING CODES

When editing text is is easy to forget about the presence of non-printing codes within the body of text. It is especially important to remember that two of these codes are added to the end of each line: one is the code for carriage return (CR) and the other a line feed (LF). Whenever the RETURN key is typed, these two codes are added to the text being edited or created; CR is added first, followed by LF. They must be considered when manipulating the pointer near or at the end of a line.

It is possible to remove only one of these codes with an editing command. For example, if the "CR" code is deleted, leaving the "LF" code, the terminal will advance to the next line, but the carriage will not move to the left-hand margin. If the "LF" code is deleted, leaving the "CR" code will return to the beginning of the line just typed thus causing overprinting. It can become very confusing determining "what happened" during an editing session if only one of these codes is inadvertently deleted.

TEXT TYPE-OUT COMMANDS

The following commands permit portions of the text in the buffer to be printed for examination. These commands do not move the buffer pointer. The Type (T, nT, HT) and Verify (V) commands are introduced.

T (Type)	Types the contents of the text buffer from the current position of the pointer through and including the next line feed character.
nT -nT ØT	Types n lines. A positive value for n causes the next n lines following the pointer to be typed. If n is negative, the n lines preceding the pointer are typed. If n is zero, the portion of the line on which the pointer is located is typed from its beginning up to and including the last character preceding the pointer's locations.
	Before issuing text deletion or insertion commands, the user must know the exact

	location of the pointer. The "ØT" command is particularly useful for determining its position. This command should be used frequently to determine that the pointer is actually located where the user expects it to be.
HT	Types the entire contents of the text buffer.
V (Verify)	Types the entire line on which the pointer is currently located.

TEXT MODIFICATION COMMANDS

Text modification commands permit the user to insert or delete text from the buffer. This collection of commands is a condensation of commands from two groups of commands as they are presented in the "TECO Manual." They are from the text insertion and text deletion command groups. Those presented here include: Kill (K, nK), Delete (D, nD), and Insert text (I text$).

K (Kill)	Deletes all the characters between the current position of the pointer and the following "CR" and "LF" codes, including the "CR" and "LF" codes.
nK -nK ØK	Performs the "K" command n times. A positive value of n causes the n lines following the pointer to be deleted. A negative value of n causes the n lines preceding the pointer to be deleted. If n is zero, text from the beginning of the line on which the pointer is located up to the pointer is deleted.
ØLK	Deletes the entire line. This is a combination of the "ØL" and "K" commands. The "ØL" command moves the pointer to the beginning of the line, and then the "K" command deletes the entire line. The "K" command will not delete text across a page boundary. If directed to do so, it will

delete text up to the boundary of the buffer and then stop.

D (Delete)

Deletes the character following the buffer pointer.

nD

-nD

Performs the "D" command n times. A positive value of n causes the next n characters following the pointer to be deleted. A negative value of n causes the n characters preceding the pointer to be deleted. If n is zero, the command is ignored.

?DTB Delete too big

The "D" command will not delete text across a page boundary. A "D" command attempting to do so results in a "?DTB" error message and the command is ignored.

Itext$ (Insert)

The letter "I" stands for "Insert," and the word "text" represents a string of ASCII characters that are inserted into the buffer at the current position of the pointer. After text insertion, the pointer is positioned immediately after the last character of the inserted text.

This is the command that allows the user to enter an entire page of material directly into

the buffer when building a file, or to insert a single character somewhere in a body of text while editing. In other words, the word "text" between the command and the "ESCAPE" ($) symbol can represent either a single character or an entire page of material.

SEARCH COMMANDS

The following commands may be used to search for a specified string of characters which occur somewhere in the input file. If the specified string of characters is found, the buffer pointer is positioned immediately after the last character in the string. The SEARCH, (S), NON-STOP SEARCH (N), SEARCH/REPLACE (FS), and NON-STOP SEARCH/REPLACE (FN) commands are introduced.

Stext$ (Search)

Searches the text buffer for the next occurrence of the character string following the current pointer position.

The word "text" between the "S" and the "ESCAPE" symbol ($) represents the string of ASCII characters for which the user wishes to search. If the string is found, the pointer is positioned immediately after the last character of the string. If it is not found, the pointer is positioned immediately before the first character in the buffer and the error message "?SRH" is printed. The word "text" represents the ASCII character string that was searched for.

?SRH Search failure "text"

-Stext$

Identical to the "SText$" command except that the search proceeds in the reverse direction. The text preceding the current pointer position is searched for the specified string of characters.

Ntext$ (Non-stop search)	This command performs the same function as the "S" command except that the search is continued beyond the present contents of the buffer (i.e., across page boundaries) if necessary. If the search reaches the end of the buffer without finding a match for the character string, it writes the contents of the buffer into the output file, clears the buffer, then reads the next page of the input file into the buffer and continues the search. If the end of the input file is reached before the character string is found, the error message "?SRH" is printed. It is then necessary to close the output file and reopen it as an input file before any further editing may be done on that file.
FStext1$text2$ (Search/Replace)	Executes a search similar to a "Stext$" search command. When the character string "text1" is found, it is deleted and then replaced with the character string "text2."

-FStext1$text2$	Performs a search similar to a FStext1$text2$ search, except that the search/replace operation proceeds backwards from the present position of the pointer.
FNtext1$text2$ (Non-Stop Search/Replace)	Executes a continuous search/replace operation through the entire file (across page boundaries).

All of the search commands execute a search by attempting to match the specified string of ASCII characters with portions of the buffer contents. They begin searching for the specified character string at the current position of the pointer. However, only the "-Stext$" and "-FStext1$text2$" commands will locate any occurrence of the character string which precedes the current pointer position. None of the commands will locate any character string which extends across a page boundary; i.e., if portions of the character string are on two different pages, none of the search commands will locate the string.

These commands are very useful for quickly moving the pointer to a desired position for editing purposes. With them the user does not need to count the number of lines or characters from the present pointer position to the desired position as he would if using the "L" or "C" commands for pointer movement. However, the user should make certain that the search terminates at the correct position and not at the end of an identical character string located in a different position in the text than the one desired. It is useful to execute a typing command after a search command to verify the location of the pointer.

COMMAND SUMMARY CHART

QUICK GUIDE TO TECO COMMANDS

Getting Started

.EDIT	This command gets the user into TECO - see the "Input/Output File Designation" section.
*	Indicates that TECO is ready to accept a command.
$	Character typed on the terminal when the ESC (escape) key is typed - used to separate and end commands.
$$	Typed after a command or command string to execute it.
RETURN = (CR)(LF)	Everytime the RETURN key is hit, a carriage return (CR) and a line feed (LF) code are added to the text.

Input/Output File Designation

All of these commands place TECO into memory and, when applicable, read the first page of the input file into the buffer.

EDIT/CREATE device:file name.extension	Creates a new file with the specifications listed.
EDIT/INSPECT device:file name.extension	Opens the specified file as the input file only.
EDIT device:file name.extension	Opens the specified file as the input file and creates a new file with the same specifications as the output file. Upon exiting from TECO, a copy of the original file is retained, but given a .BAK extension.
EDIT/OUTPUT:device:file name.extension (output file) device:file name.extension (input file)	Opens the specified files as the input and the output files. If the two file specifications differ - the input file is retained; if they are the same - the input file is deleted upon exiting from TECO.

Reading File Contents into the Buffer

P	(Page) Writes the present contents of the buffer into the output file, clears the buffer, reads the next page of the input file into the buffer, and positions the pointer at the beginning of the buffer.

Moving the Pointer

J	(Jump) Moves the pointer to the beginning of the buffer.
ZJ	(Jump to the End) Moves the pointer to the end of the buffer.
L	(Line) Moves the pointer to the beginning of the next line.
nL	Moves the pointer to the beginning of the "nth" line. Positive value of n - move forward; negative - move backwards; Ø - move to the beginning of the line on which the pointer is currently located.
C	(Character) Moves the pointer ahead by one character.

nC	Moves the pointer n characters from its present position. If n is positive - move forward; negative - move backwards; Ø - no movement.
R	(Reverse) Reverses, or backs up, the pointer across one character.
nR	Executes the "R" command n times.

Typing out the Buffer Contents (pointer does not move)

HT	Types out the entire contents of the buffer.
T	(Type) Types the current line, from the present position of the pointer to the end of the line.
nT	Types n lines before or after the line on which the pointer is currently located. If n is positive - type n lines following the pointer; negative - n lines preceding; Ø - type from the beginning of the line on which the pointer is located up to and including the character preceding the location of the pointer.
V	(Verify) Types the entire line on which the pointer is currently located.

Searching for a String of Characters

SText$	(Search) Searches from the current pointer position to the end of the buffer for the specified character string (text).
Ntext$	(Non-Stop Search) The same as Stext$ except that the search continues to the end of the input file if necessary.
FStext1$text2$	(Search/Replace) Performs a Stext$ search for "text1" and replaces it with "text2."

For all three commands, the pointer is positioned after the last character in the string if the search is successful.

Inserting and Deleting Text

Itext$	(Insert) Inserts the specified text at the current pointer location. After text insertion, the pointer is located after the last character inserted.
K	(Kill) On the line containing the pointer, deletes all the characters following the pointer up to and including the next "CR" and "LF."
nK	Executes the "K" command n times. If n is positive - delete n lines following the pointer; if negative - n lines preceding; Ø - delete all characters preceding the pointer on the line which currently contains the pointer.
D	(Delete) Deletes a single character following the pointer.
nD	Deletes n characters. If n is positive - delete n characters following the pointer; negative - n characters preceding; Ø - no effect.

Exiting from TECO

EX	(Exit) Takes the user out of TECO and returns him to monitor command level.

TECO EXAMPLES

The TECO program will be used in this section to create and then edit a file containing X-ray diffraction data. The following examples are specifically tailored for use at the USGS facility. If the user plans to recreate the examples as they are presented, diskette #1 (MCØ: Master Diskette) must be in unit MCØ: and a blank diskette (or the one used in earlier examples) in drive unit MC1:. The examples are presented with the same format that is used for the monitor command language examples.

1. Creating a file with TECO.

The following describes how to create a new file. In this example, a data file which consists mostly of numerical data is created. However, the same instructions apply to any type of ASCII file.

At this point it is assumed that the RT-11 operating system has been invoked (boot-strapped) and the monitor is ready to accept a command.

```
.EDIT/CREATE MC1:CUBIC.DAT↓
```

The RT-11 "EDIT" command is used to load the TECO.SAV file into memory. Since the user is creating a new file, the "/CREATE" option is used to open a new file named CUBIC.DAT, which is located on the MC1: storage device. This file will serve as the output file for all text entered into the buffer.

As soon as the "EDIT" command is executed, an "*" appears indicating that TECO is in memory and ready to accept a command.

```
*
```

```
*ITITLE:URANINITE CUBIC SYSTEM TEST DATA)
SYSTM:CUBIC)
A    :  5.3700)
THEMX: 61.00000)
PWL  :  1.54178)
DSPAC:  1      3.090 1 1 1)
DSPAC:  2      2.686 0 0 0)
DSPAC:  3      1.900 0 0 0)
DSPAC:  4      1.620 0 0 0)
DSPAC:  5      1.540 0 0 0)
$
```

After the asterisk, the "I" (insert)

```
*I
```

command is used to let TECO know that everything following the "I" is to be placed directly into the buffer beginning at the current location of the pointer. Since TECO has just been loaded into memory, the pointer is at the beginning of the buffer.

Between the "I" and the next "$" are the contents of the file being created. The user should try to copy this file exactly as it is presented. Especially pay attention to the location of spaces and the alignment of entries within a column.

If the data listed in a file such as this is to be used by a particular program, it must be arranged in the exact format required by the program specifications. If this format is not followed exactly, incorrect data may be transferred to the program from this file. Incorrect formatting may cause the program to abort when it attempts to use this data.

```
EX$$
```

After the ESCAPE symbol ($) terminating the insert command, the "EX" command is entered. Both the exit and insert commands are then executed by typing ESC twice ($$). "EX" is used to exit

from TECO and return to the RT-11 monitor command level. However, before this happens, all the text in the buffer is written to the file CUBIC.DAT and the file is closed.

```
.DIR MC1:CUBIC.DAT↵
 01-Jan-82
CUBIC .DAT      1 01-Jan-82
 1 Files, 1 Blocks
 659 Free blocks
```

A return to monitor command level is indicated by a dot typed on the left margin. Use the "DIR" command to confirm the existence of the file just created.

```
.TYPE MC1:CUBIC.DAT↵

TITLE:URANINITE CUBIC SYSTEM TEST DATA
SYSTM:CUBIC
A     :  5.3700
THEMX: 61.00000
PWL   :  1.54178
DSPAC:  1      3.090 1 1 1
DSPAC:  2      2.686 0 0 0
DSPAC:  3      1.900 0 0 0
DSPAC:  4      1.620 0 0 0
DSPAC:  5      1.540 0 0 0
```

The "TYPE" command is used to confirm that this file contains all the information that was entered into it with TECO.

Notice that the first line of the file begins on the left margin, not three characters to the right of the margin where typing began during creation of the file. The asterisk and the "I" command, which occupied the first and second spaces, were not entered into the buffer.

2. Editing a file with TECO

Now that a file has been created, the user may want to correct errors, add additional data, or delete data from the file. All of these operations can be easily performed with TECO editing commands. A file named CUBIC1.DAT on the MCØ: storage device will be used for this example. It contains the same data as CUBIC.DAT, but with numerous errors. In this example, long command strings are repeatedly used. At first they may seem cumbersome to the user, but by the end of the exercise the user should feel comfortable with them.

In several cases the most efficient method of correcting a particular error has not been used. This has been done on purpose so that the user can gain experience with a large number of commands and manipulating the pointer.

```
.EDIT MC0:CUBIC1.DAT)
```

Load TECO into memory using the RT-11 EDIT command. None of the options are required to simply edit a file. TECO then retrieves the previously created CUBIC1.DAT from the MCØ: storage device. CUBIC1.DAT is now designated as both the input and output files, and a copy of the original file (unedited) will be retained as a .BAK file after the user exits TECO.

Use of the "EDIT" command also results in the first page of the input file (in this case the entire file) being read into the buffer.

As soon as the "EDIT" command is executed, an asterisk is typed at the left hand margin to indicate that TECO is ready to accept commands.

```
*HT$$
```

To view the contents of the buffer (and in this case the entire file) "HT" is used, which requests that the entire contents of the buffer be typed out.

This command is then executed by typing the ESC key twice, indicated by the two $ after the "HT".

Once the "HT" command has been executed, the contents of CUBIC1.DAT

```
TITKE:URANINITE CUBIC SYSREM TEST DATA
SYSTM:CUBIC

A    :  5.3700
THEMX: 61.50000
PWL  :  1.54178
DSPAC:  1     3.090  1  1  1
DSPAC: 2     2.686  0  0  0
DSPAC:  2     1.900  0  0  0
DSPAC:  4     1.630  6  7  4
DSPAC: 5     1.540  0  0  0
```

are typed out. The contents of this file are the same as the contents of CUBIC.DAT except that this one contains eleven errors. The eleven errors are marked on a copy of the file listed below.

```
Line
Number.
 1. TITKE:URANINITE CUBIC SYSREM TEST DATA
 2. SYSTM:CUBIC
 3. (blank line)
 4. A    :  5.3700
 5. THEMX: 61.50000
 6. PWL  :  1.54178
 7. DSPAC:  1     3.090  1  1  1
 8. DSPAC: 2     2.686  0  0  0
 9. DSPAC:  2     1.900  0  0  0
10. DSPAC:  4     1.630  6  7  4
11. DSPAC: 5     1.540  0  0  0
```

The procedure for correcting each mistake is given below. In general, each mistake is corrected by using one command string. So that the individual commands can be identified more easily, each command string is broken down into its component commands, separated by spaces, at the beginning of each subsection.

To correct the misspelled word "TITKE" on line 1.

TITKE:URANINITE.....

Command string: 3C D IL$ V $$

Before issuing any commands, the current position of the pointer must be ascertained. Since the "P" command was used earlier, the buffer pointer is located at the beginning of the buffer.

Use the "C" command to move the pointer to the right by three characters (3C) so that is is between T and K.

TIT↑KE:URANINITE.....

Delete the next character (K) with a single "D" (Delete) command. Now insert the letter L in this position with the "I" (Insert) command, which always must be terminated by typing a single ESC. Finally, to check the results of this command string, use the "V" (Verify) command to have the entire line typed out. A double ESC executes the command string.

*3CDIL$V$$

TITLE:URANINITE CUBIC SYSREM TEST DATA

(2) To correct the misspelled word "SYSREM" on line 1.

...CUBIC SYSREM TEST DATA

Command string: SSYS$ D IT$ V $$

*T$$

E:URANINITE CUBIC SYSREM TEST DATA

To find the current location of the pointer use the "T" (Type) command, which types the line on which the pointer is located, beginning at the current position of the pointer. The portion of the line typed out indicates that it is located between the letters L and E in the word TITLE.

TITLE: URANINITE.....

This indicates that after the "I" command is used, the pointer is located directly after the inserted characters.

The "S" (Search) command instructs TECO to search for the character string SYS and then place the pointer after the string. This will place the pointer between the letters S and R in the misspelled word "SYSREM."

...CUBIC SYSREM TEST DATA

*SSYSDITV$$

TITLE:URANINITE CUBIC SYSTEM TEST DATA

The "D" command deletes the next letter after the pointer (R) and then the "I" command is used to insert the letter T in its place. To check the results, the "V" command is used to have the entire line typed out.

The first line is now correct.

(3) To remove the blank line after line 2.

Command string: 2L K -L 2T $$

Use the command "2L" (Line) to advance the pointer by two lines. Whenever the "L" command is used, the pointer is positioned at the beginning of the line to which it is instructed to go. The pointer is now at the beginning of the blank line.

The "K" (Kill) command deletes the entire line. To check the results, the pointer is moved back by one line (-L) and then the command "2T" is used to have the next two lines following the current pointer position typed out.

*2LK-L2T$$

SYSTM:CUBIC
A : 5.3700

The typed lines reveal that the blank line has been removed.

(4) To change the number 61.5 on line 5 to 61.0.

THEMX: 61.50000

Command string: FS61.5$61.0$ V $$

To easily change the number 61.5 on line 5 to 61.0, the search/replace command, FS61.5$61.0$, is used. The number 61.5 is searched for and then replaced with the number 61.0. The "V" command is used to check the results.

*FS61.5$61.0$V$$

THEMX: 61.00000

After execution of this command, the pointer is located after the first zero in the number 61.00000.

```
THEMX:  61.00000
              ↑
```

(5) To move all the numbers on line 8 to the right by one space.

```
DSPAC: 2    2.000  0  0  0
```

```
Command string:  3L 6C I $ -L 3T $$
```

The pointer is currently on the line just corrected, "THEMX: 61.0000," which is line 5. By using the "3L" command it will be positioned at the beginning of line 8.

```
↑DSPAC:  2 . . . .
```

If a single space is inserted between the colon and the number 2, the entire portion of the line following this point will be moved to the right by a single space. The command "6C" moves the pointer to the proper position.

```
DSPAC:  2....
        ↑
```

A blank is then inserted with the "I" command. To check that the numbers in this line are now aligned with those of the lines above and below, the "-L" command is used to move the pointer back by one line (to the beginning of

```
*3L6CI $-L3T$$

DSPAC:  1     3.090  1  1  1
DSPAC:  2     2.686  0  0  0
DSPAC:  2     1.900  0  0  0
```

line 7) and then the next three lines are typed out with the "3T" command.

(6) To replace the number 2 following the colon in line 9 with the number 3.

```
DSPAC:  2     1.900  0  0  0
```

Command string: 2L 8C D I3$ V $$

First, check the current location of the pointer with the "V" command.

```
*V$$

DSPAC:  1     3.090  1  1  1
```

This indicates that the pointer is on line 7, so the "2L" command is used to place it at the beginning of line 9.

```
DSPAC:  2     1.900....
        ↑
```

It is then moved to the right by eight characters (8C).

```
DSPAC:  2     1.900....
           ↑
```

The following character (2) is then deleted (D) and replaced with the number 3 (I3$). The "V" command is then used to check the results.

```
*2L8CDI3$V$$

DSPAC:  3     1.900  0  0  0
```

(7) To change the number 1.630 on line 10 to the number 1.620.

```
DSPAC:  4     1.630  6  7  4
```

Command string: S1.6$ D I2$ ØT $$

The search command, S1.6$, is used to position the pointer between the characters 6 and 3.

DSPAC: 4 1.630 6 7 4

The 3 is then deleted and replaced with a 2. To both check the results and locate the pointer, the "ØT" command is used. When this command is executed the line on which the pointer is located is typed out up to the character preceding the pointer. The portion of the line typed out reveals that the pointer is in the position indicated below.

*S1.6$DI2$0T$$

DSPAC: 4 1.62

DSPAC: 4 1.620 6 7 4

(8), (9), (10) To change the last three numbers on line 10 to zeros.

DSPAC: 4 1.620 6 7 4

Command string: 3C D IØ$ 2C D IØ$
2C D IØ$ V $$

Now that the user knows that the pointer is located between the characters 2 and Ø on line 10, he can use the "3C" command to move it just to the left of the number 6.

DSPAC: 4 1.620 6 7 4

This number is deleted (D) and a zero

is inserted in its place (IØ$). The pointer is then advanced two characters (in this case they are blanks) to the right (2C) and the number 7 is changed to Ø (IØ$).

```
DSPAC:  4     1.620  0  7  4
DSPAC:  4     1.620  0  0  4
```

The number 4 is similarly changed to Ø. Finally, the results are checked with the "V" command.

```
*3CDI0$2CDI0$2CDI0$V$$

DSPAC:  4     1.620  0  0  0
```

(11) To move the number 5 in the last line to the right by one space.

```
DSPAC:  5     1.540  0  0  0
```

Command string: L 6C I $ 2C D V $$

First, the pointer is moved from its current position on the second to last line to the beginning of the last line by using the "L" command. It is then advanced six characters to the right (6C) which places it directly after the colon.

```
DSPAC:  5     1.540  0  0  0
```

A blank is inserted here (I $), which will cause the portion of the line to the right of this point to be moved to the right by a single space (as with mistake #5). Since the user wishes to move only the number 5 to the right, a

space must be removed after the number 5. This will bring the portion of the line following the number 5 back to its original position. Advancing the pointer by two characters (2C) brings it to the right of number 5.

```
DSPAC:  5      1.540  0  0  0
```

```
*L6CI $2CDV$$
DSPAC:  5      1.540  0  0  0
```

The character following 5, which is a space, is then deleted (D). The results are then checked with the "V" command.

The "HT" command is then used so that the entire file is typed out for inspection.

```
*HT$$

TITLE:URANINITE CUBIC SYSTEM TEST DATA
SYSTM:CUBIC
A    :  5.3700
THEMX: 61.00000
FWL  :  1.54178
DSPAC:  1      3.090  1  1  1
DSPAC:  2      2.686  0  0  0
DSPAC:  3      1.900  0  0  0
DSPAC:  4      1.620  0  0  0
DSPAC:  5      1.540  0  0  0

*EX$$
```

The user may now return to monitor command level with the "EX" command.

Once this command is executed, a copy of both the original file and the edited version will be on the MCØ: storage device. The original file is labeled CUBIC1.BAK and the edited file is labeled CUBIC1.DAT.

```
.DIR MC0:CUBIC1.DAT,MC0:CUBIC1.BAK

 01-JAN-82
CUBIC1.DAT       1 01-JAN-82
CUBIC1.BAK       1 01-JAN-82
 2 Files, 2 Blocks
 12 Free blocks
```

When the computer returns to the RT-11 command level, the user may use the "DIR" (DIRECTORY) command to verify that these files exist. The message typed out reveals that both files exist and are on the MCØ: storage device.

```
.TYPE MC0:CUBIC1.BAK
TITKE:URANINITE CUBIC SYSREM TEST DATA
SYSTM:CUBIC

A    :  5.3700
THEMX: 61.50000
PWL  :  1.54178
DSPAC:  1     3.090  1  1  1
DSPAC: 2     2.686  0  0  0
DSPAC:  2     1.900  0  0  0
DSPAC:  4     1.630  6  7  4
DSPAC: 5     1.540  0  0  0
```

```
.TYPE MC0:CUBIC1.DAT
TITLE:URANINITE CUBIC SYSTEM TEST DATA
SYSTM:CUBIC
A    :  5.3700
THEMX: 61.00000
PWL  :  1.54178
DSPAC:  1     3.090  1  1  1
DSPAC:  2     2.686  0  0  0
DSPAC:  3     1.900  0  0  0
DSPAC:  4     1.620  0  0  0
DSPAC:  5     1.540  0  0  0
```

To verify that CUBIC1.BAK contains the original (unedited) file and that CUBIC1.DAT contains the edited file, use the "TYPE" command to check the contents of each file.