RTE-6/VM
Interactive Line Editor/EDITR

Reference Manual

HEWLETT
PACKARD

HEWLETT-PACKARD COMPANY
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

MANUAL PART NO. 92084-90017
Printed in U.S.A. December 1981

PRINTING HISTORY

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates, however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

To determine what software manual edition and update is compatible with your current software revision code, refer to the appropriate Software Numbering Catalog, Software Product Catalog, or Diagnostic Configurator Manual.

First Edition ..... Dec 1981

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Preface

This reference manual describes the use of the Interactive Line Editor, EDITR. EDITR is included in and is a standard feature of the Hewlett Packard RTE-6/VM operating system. EDITR is primarily intended for use in a Batch; a Multi-User or Multi-Terminal Monitor; a MULTIPOINT environment; or on a DS/1000-IV network node.

EDITR is used to create and edit ASCII files. This manual provides a complete description of EDITR commands and functions with indications of operating systems considerations, examples, and references to other RTE system manuals for additional information.

EDITR is an interactive line editor, while EDIT/1000 is both a line and screen editor.

If you want to use EDITR for simple editing tasks, read Chapters 1 and 2. In this case, the important things you need to know about RTE or the File Manager is how you want to initiate and terminate your editing sessions.

This manual consists of the Preface, 3 Chapters and Appendix A:

Chapter 1 - Introduces EDITR; describes how to initiate interactive and batch jobs; describes keyboard conventions and display formats; describes sequence of decisions to be made about the EDITR environment; and explains on-line loading of EDITR.

Chapter 2 - Lists and describes EDITR commands with selected examples of their use.

Chapter 3 - Discussion of EDITR in special situations.

Appendix A - EDITR Messages.

Additional publications which should be available for reference when using this manual are:

92084-90005	RTE-6/VM Programmer's Reference Manual
92084-90006	RTE-6/VM Batch and Spooling Reference Manual
92084-90004	RTE-6/VM Terminal User's Reference Manual
91730-90002	Multipoint Terminal Interface Subsystem User's Guide
91750-90002	DS/1000-IV User's Manual

Table of Contents

Chapter 1 General Information

Chapter 2 EDITR Operations

List of Illustrations

List of Tables

Chapter 1
General

Introduction

The RTE Line Editor (EDITR) is commonly used for:

* Line Editing using local,remote, or multipoint devices
* Creation of new programs or data files in ASCII code.
* Modification of new or existing ASCII files.
* Merging several ASCII files into a single file

EDITR is a line editor used either in interactive or batch mode under RTE-6/VM. When used interactively, EDITR accepts operator commands from a keyboard device. When in batch mode, EDITR commands are included in the job command file. For additional information on batch processing, refer to Chapter 3 and the RTE-6/VM, Batch and Spooling Reference Manual, HP Part No. 92084-90006.

EDITR Word Areas

Basic Concept

EDITR references a "source file" and an "output file" and uses two temporary work areas on disc (see Figure 1-1 for an overview). During editing, the user called (original) source file is copied into a temporary work area on disc called the "source work area". In understanding the EDITR algorithm, it is helpful to remember that user entered EDITR commands work only on the copy in the "source work area" while the original text remains in its original state and location on disc.

The editing process begins with user entry of EDITR commands, which work on text in the "source work area". As each line of text is reviewed, edited, or passed by the user it moves to another temporary disc work area called the "destination work area".

An editing pass is completed when EDITR has read and passed text into the "destination work area". Upon completion of an editing pass, the "source work area" is replaced by the current "destination work area" and a new "destination work area" is set up for continued editing activity. This process is continuously repeated until the user is satisfied that the text is correct and enters a Terminate Command. In selecting the Terminate Command, the user has three choices which are discussed (with examples and comments) later in this chapter. Briefly, these are to abort EDITR (A); update the original source file to incorporate all edited text and exiting the EDITR (ER); or leave the original source in its original state while creating a new, separate source file containing the just edited text and exiting EDITR (EC). The appropriate selection of EDITR Terminate Command will largely depend upon user objectives (application oriented). Execution of either an ER or EC command will cause the updated text to be written into the output file.

While this sounds complex and difficult to use, in actual use, the swapping of the "source work" and "destination work" areas is efficiently managed by EDITR while letting the user concentrate on text editing with the full confidence that original source is always available.

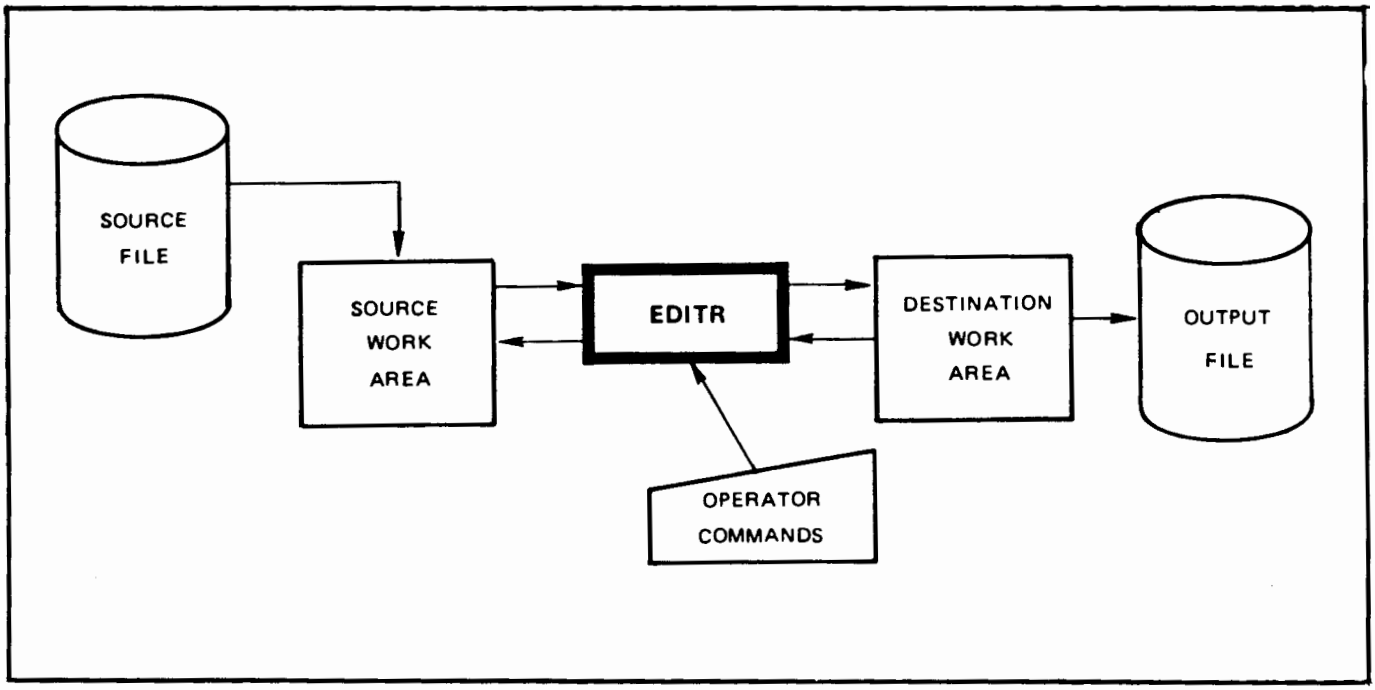


Figure 1-1. File/Work Area Relationship

Pending Line

When EDITR is run, and the source file (named at the beginning of the editing session) is read into the source work area, the first line is displayed on the terminal. This displayed line is the current line available for editing and is called the "pending line". This line remains the pending line until the user requests a new pending line with an EDITR command. In this way the user can continue to re-edit the same line until it is correct and a new line is requested.

When the user requests a particular line of text, EDITR searches through the source work area until the requested line is encountered. That line then becomes the new "pending line" and is displayed on the terminal. EDITR maintains a "pending line" pointer into the source work area.

When the new pending line is displayed, the old pending line, and all other lines passed over by the pointer in the search for the new pending line, are written to the destination work area (see the Jump command, Chapter 2, for an exception), and are accessible when the "destination work area" becomes the "source work area", i.e. the current editing pass is completed. In other words, when the user goes from line 0028 back to line 0024 the source work area is replaced by the destination work area. Typically the user is not concerned with this exchange because it is managed by EDITR. However, the user does need to remember that once the "source work area" is replaced, the previous "source work area " is gone along with any lines deleted or changed in the last pass of EDITR. Any changes entered into the former "source work area" are no longer accessible by their former line numbers. For example, if line three is deleted in an editing pass, the original line four becomes the new line three. EDITR, also, permits lines of text to be accessed without using line numbers.

Keyboard Conventions

Keyboard commands are used to direct EDITR to do replacements, insertions, deletions, searches, and exchanges of text. These functions can be performed on characters within a line, they can be used to manipulate one entire line, or they can be used on groups of lines.

To use EDITR efficiently, the user must know the command syntax and keyboard conventions it expects. The command syntax required by EDITR is summarized in Table 1-1. For keyboard conventions, check the appropriate manual for the terminal being used. If a Multipoint terminal is used, consult Chapter 3 and the Multipoint Terminal Interface Subsystem User's Guide, Hewlett Packard P/N 91730-90002.

Conventions Used in Examples:

{ } Space entered at the keyboard.

CNTL/X Indicates a nonprinting control character (in this case, X) entered at the keyboard but not visible on the screen.

Table 1-1. Command Syntax

CONVENTION	MEANING	COMMAND	EXAMPLE	COMMENTS
UPPER CASE LETTERS	Literals that must be specified as shown	F	F	No Parameters
lower case letters	Variables to be replaced by values as defined in text	Wa,b	W7,9	Constants 7 and 9 replace variables a and b
CNTL/	Nonprinting control characters entered by pressing the CNTL key and another key simultaneously	CNTL/key	CNTL/C	
[ ]	Bracketed parameters are optional; if omitted, default values are supplied	filename[:sc[:crn]] example: MYFILE:AA:2		

Display Formats

The pending line displayed and the listings output by EDITR are always preceded by two blanks. This convention allows room for the EDITR prompt and a single character command, and results in the new text being automatically aligned with that displayed by EDITR.

Error messages, in contrast, always begin in column 1. This is so that the difference between an EDITR message and a line of text can easily be seen:

```
  EOF     EDITR text
EOF       EDITR message
```

Calling EDITR

The user can call EDITR with either the File Manager or System RU commands :

```
:RU,EDITR [,LU [,line length ]]
OR
*RU,EDITR [,LU [,line length ]]
```

These commands are described in the RTE-6/VM TERMINAL USER'S REFERENCE MANUAL, HP Part No. P/N 92084-90004.

The two optional parameters passed to EDITR when it is scheduled are:

LU — The logical unit may specify any device upon which the user can enter EDITR commands. The default convention is the user's terminal. If the user is editing in batch mode as described in the EDITR IN BATCH ENVIRONMENT section, this device could be a read only device (such as a card reader). If, using Batch with Spooling, this device could be LU 5 (command input) or another LU corresponding to a spool file containing the commands.

LINE LENGTH — The line length is an integer number which specifies the number of characters in a line. The user may specify a line length compatible to his output device. Any line longer than 150 characters (or the device specified length) will be truncated. Table 1-2 lists the device specified output commonly used with EDITR.

For example, to schedule EDITR from the File Manager with commands input from the user's terminal with lines limited to 72 characters, the command would be:

```
:RU,EDITR,,72
```

EDITR Prompt Character

When EDITR is used in the interactive mode, it prompts for input with a slash (/). The X command may be used to change this prompt character to any other selected character. Thereafter, this specified character will be output as the EDITR prompt. See X Command in Chapter 2 for information on changing EDITR prompt.

Table 1-2. Maximum Line Lengths for Common Devices

DEVICE	LINE LENGTHS	COMMENTS
Punched Cards	80 Characters	
CRT	72 - 80 Characters	
Magnetic Tape	150 Characters	Default maximum length supported by EDITR.
TTY Device	72 Characters	Longer lines will encounter printing problems.
Paper Tape	150 Characters	Default maximum length supported by EDITR.
Disc	150 Characters	Default maximum length supported by EDITR.
Line Printer	80 - 132 Characters	Number of characters varies with printer model. Consult appropriate manual for your printer.

File Editing

File Definition

The NAMR used by the EDITR is a simplified version of the File Manager NAMR. The EDITR version of NAMR is defined to be a file name followed by two subparameters. The subparameters may be omitted from the end of the list. If an embedded subparameter is omitted, its position may be indicated by the colon (:). The NAMR format is :

```
filename [:security code [:cartridge reference number ]]
```

Refer to the RTE-6/VM Terminal User's Reference Manual, HP PART No. 92084-90004, for an explanation of NAMR parameters and files.

Edit Existing File

When EDITR begins execution, it requests information about the file to be edited and prompts the user.

```
SOURCE FILE?
/
```

At this point, there are three legal responses the user can make, each of which must be followed by a carriage return:

1. 0 (zero) — If "0" is specified, the user will start working with an empty file. The EDITR commands can then be used to enter and edit lines of text. When creating a file, this is the recommended response. See the CREATE FILE WITH EDITR (next section) for an example.

2. : (colon) — If ":" is specified, the EDITR is immediately aborted and control returns to the program from which the EDITR was scheduled.

3. filename — If a file name is specified, the contents of the file will be copied into the EDITR's source work area. The number of characters per line in the named file cannot be greater than the current maximum line length.

The file may be a Type 0 file to read source information directly from a peripheral device. The file name may be specified with or without a security code and a cartridge reference number. For example, to specify a file without a security code, but with a cartridge reference number:

```
/FILE1::27
```

Refer to RTE-6/VM Terminal User's Reference Manual, Hewlett Packard P/N 92084-90004 for an explanation of type 0 files.

Create File With EDITR

To create a file with EDITR, enter a 0 (zero) in response to the EDITR prompt "SOURCE FILE?". EOF is printed and the following lines can be entered as illustrated:

```
:RU,EDITR
SOURCE FILE?
/0  <------------Enter 0 to put empty file into EDITR's
                 source work area.
EOF
/ NEW FILE  <----Enter space to add line of text following
                 pending line, then enter line of text to
                 be added.
/ LINE TWO
/ LINE THREE
/ECFILE1 <-------Exit EDITR and create FMP file named FILE1.
```

The above command sequence will enter the three lines of text shown into the newly created file named FILE1.

EDITR Termination

The EDITR terminate commands (EC and ER) assign the final version of text in the destination work area to a File Manager type 4 file. The destination work area can also be output directly to a device through a type 0 file.

EDITR automatically truncates trailing blanks when it first moves a source file into the work area. Therefore, an existing file does not have trailing blanks when it is replaced.

Loading EDITR Software

EDITR is a utility program that may be loaded on-line using the RTE Relocating Loader (LOADR) or generated into the RTE operating system. EDITR requires a minimum partition size of 8 pages, although 9 pages is recommended for adequate internal buffer space allocation. There are no special libraries required, just the system libraries already generated. To save EDITR as a Type 6 File, use the FMGR SP command. EDITR should be specified as a Type 2 or 3 Program when it is loaded on-line or during generation. A sample loader command file is :

```
SZ,9
RE,%EDI6R
END
```

To generate EDITR into the operating system, the following entries should be included in the answer file :

```
PROGRAM INPUT PHASE
REL,%EDI6R

PARTITION DEFINITION PHASE
*MODIFY PROGRAM PAGE REQUIREMENTS?
EDITR,9
```

EDITR must be used with the following terminal drivers :

DVR00
DVR05/DVA05
DVR07

For additional information on Generation or Loader considerations consult the RTE-6/VM System Manager Reference Manual, HP Part No. 92084-90009.

Chapter 2
EDITR Operations

EDITR Commands

This is a brief overview of EDITR functional groups and related commands:

* CONTROL Commands — Provides additional EDITR features beyond the usual text manipulation functions.

* DISPLAY Commands — Causes the contents or information about the contents of the source work area to be displayed. The display normally occurs on the user's screen but commands are provided to allow display on other logical units or to dump to the printer.

* LINE EDIT Commands — Allows user to manipulate one entire line of text at a time. In addition to these EDITR commands, certain keys on the keyboard (such as DEL) can, also, be used.

* DEL Key — Used to delete a line, if pressed before the the RETURN key enters the line into the destination work area. It prints a back slash (\), and causes a line feed and carriage return to the start of the next line (prompt is not not repeated) where the correct line can be entered.

* CHARACTER EDIT — Provides a way to change the contents and modify the structure of the current line of text. Any of four commands allow the replacement, insertion and deletion of characters. Each of these commands uses nonprinting control characters so that alignment is maintained in the text. Each must be used with an initial "P", "C" or "O" command.

The current delimiter is used as a "place holder" to preserve existing text in the pending line. The P, C and O commands are not themselves character edit functions. They are used to determine the line disposition during the edit: P leaves the edited line as the pending line; C advances the pending line to the next line after the edit; O sends the pending line to the destination work area and then edits a copy of the line and leaves it pending.

The Character Edit functional group is inoperative in Multipoint Environments.

* PATTERN EDITS

Typically involves multiple lines of text from the source work area. There are two kinds of pattern edits: "searches" and "exchanges".

A search matches a character string called a "find field" with a corresponding string in the source work area. There are four search commands (B, D, F and J) which differ in where they begin the search and the length of the data block which they search.

Exchanges consist of two sets of character strings entered at the keyboard. When the first string is encountered in the source work area, it is replaced by the second string. The commands used to perform exchanges are: G, Y, X, Z, V and U.

* TERMINATE

Used to end EDITR operations. With the exception of the Abort command, they perform any enabled exchanges, cause data remaining in the source work area to go to the destination work area and cause the edited destination work area to be written to the output file.

All of the normal EDITR termination commands begin with E. Once E is entered, EDITR begins its termination process. Only the second letter of a normal terminate command or DEL can be entered at this time.

Output of a Type O file to a device can only be made using the ER terminator because EDITR will not create a Type O file using an EC command.

Table 2-1 EDITR Command Summary

FUNCTION	COMMAND	DESCRIPTION	Page No.
CONTROL	CNTL/G	Bell Control	2-6
	K	Kill Trailing Blanks	2-14
	M	Merge Source File Following Pending Line	2-16
	T	Set Tab Stops	2-7
	W	Set Window	2-9
	X	Change EDITR Prompt Character	2-5
	#	Sequence Numbers	2-10
	=	Set Line Length	2-12
DISPLAY	H	Display the Number of Characters in the Pending Line	2-26
	HL	Display Header	2-28
	L	Display a Number of Lines	2-18
	n	Display Specified Line	2-20
	N	Display Pending Line Number	2-23
	ND	Display the Number of Characters in the Destination Work Area	2-24
	P	Display Pending Line	2-32
	S	Display Approximate Number of Words in Destination Work Area	2-31
	^	Back up in Destination Work Area	2-29
	/ or +	Space Down a Number of Lines and Display	2-21
LINE EDITS	C	Edit Pending Line and Advance Line	2-34
	I	Insert Text Before Pending Line	2-38
	O	Duplicate Pending Line and Edit	2-35
	P	Edit and Display Pending Line	2-32
	R	Replace Pending Line with Text	2-37
	{ }	Insert Text after Pending Line	2-39
	-	Delete a Number of Lines	2-40

Table 2-1 (Cont'd). EDITR Command Summary

FUNCTION	COMMAND	DESCRIPTION	Page No.
CHARACTER EDITS *	CNTL/C	Cancel Characters	2-45
	CNTL/I	Insert Characters	2-43
	CNTL/R	Replace Characters	2-41
	CNTL/S	Insert Characters	2-43
	CNTL/T	Truncate Remainder of Pending Line	2-46
* Inoperative on Multipoint Terminals			
PATTERN EDITS	Search Commands		
	B	Find a Line with Find Field SOF to EOF	2-47
	CNTL@	Find a Zero Length Line Within This Field	2-47
	D	Delete Lines to Find Field or EOF	2-53
	F	Find a Line with Find Field Pending Line to EOF	2-50
	J	Jump to Find Field and Make It New Pending Line	2-58
	;	Find Tab Field	2-47
	esc	Find Field of Indefinite Length	2-47
	/	Find Field Within Window	2-47
	Exchange Commands		
	G	Character Replace on Pending Line	2-61
	U	Unconditional Exchange (no list)	2-74
	V	Unconditional Exchange (list)	2-71
	X	Enable Exchange Pattern All Lines (list)	2-66
	Y	Exchange on Pending Line, Display Next Occurrence of Pattern	2-63
	Z	Enable Exchange Pattern for Next Edit (no list)	2-68
TERMINATE	A	Abort EDITR	2-77
	ECnamr	Create File Manager File	2-78
	ER	Replace Old File with New File Do Not Change Name	2-79
	ERnamr	Replace Old File with New File Change Name	2-79

X (Change EDITR Prompt Character)

Changes the EDITR prompt character to a user defined prompt character.

Xx
x New prompt character (default is slash).

EXAMPLE:

```
/X$     <-----Changes the prompt from a slash (/) to a dollar sign
              ($).
$       <-----New EDITR prompt.
```

COMMENTS:

If the prompt character is changed, then the new prompt is required as the delimiter between the fields of an exchange command (i.e., for X, Y etc. commands) and for each character to be preserved when doing a character edit (i.e., in P, C, and O commands).

The space down command (/) is not affected by changing the prompt character.

The default for the EDITR prompt is always a slash at the start of an edit session regardless of what it might have been changed to in a prior edit session.

CNTL/G (Bell Control)

Turns terminal bell on or off. When EDITR is scheduled, on a terminal with a bell, the bell is rung automatically every time a prompt is displayed.

CNTL/G
A control G is input by striking the "G" key while pressing the "CNTL" key on the terminal. It is a nonprinting character.

COMMENTS:

The bell control only applies to the period within an edit session. The bell is always on when the edit session starts, even if it had been turned off in a prior session and not turned back on.

T (Set Tab Stops)

Changes the EDITR tab character (;) and the default tab stops (7th and 21st columns) to user defined values.

```
+-----------------------------------------------------------------+
|                                                                 |
| Tx                  <---Change tab control character, leave stops.
|                                                                 |
| Tts1,s2,...,s10     <---Change tab stops, leave control character.
|                                                                 |
| Txs1,s2,...,s10     <---Change tab stops and control character.
|                                                                 |
|-----------------------------------------------------------------|
|                                                                 |
| x                                                               |
| New tab control character (replaces the original semicolon or   |
| current tab control character).                                 |
|                                                                 |
| t                                                               |
| Current tab control character.                                  |
|                                                                 |
| s1,s2,...,.s10                                                  |
| New column numbers of the tab stops (replacing default values of
| 7 and 21).  The maximum number of tab stops that can be defined at
| one time is 10.                                                 |
|                                                                 |
+-----------------------------------------------------------------+
```

EXAMPLES:

1.

```
  /T%4,9        <---Changes the tab character to a percent sign (%)
                    and changes the tab stops to columns 4 and 9.
  / %A%B        <---Command to add a line with an "A" in column 4 and
                    a "B" in column 9.
  /P            <---Command to display pending line.
       A    B  <---Displayed pending line showing "A" in column 4 and
                    "B" in column 9.
```

2.

```
  /T;11,22      <---Changes tab stops to columns 11 and 22 without
                    changing tab control character from semicolon (;).
```

COMMENTS:

Tabs used beyond the highest defined stop are replaced with blanks. For instance, using the above example:

```
/ %A%B%C%D        <---Command to add line with letters A, B, C, and
                      D at tab stop locations.
/P                <---Command to display pending line.
     A    B C D  <---Displayed pending line showing "A" in column
                      4 and "B" in column 9 (the defined tab stops).
                      Since no tab stops have been defined beyond
                      column 9, "C" and "D" follow with one blank
                      space preceding each of them.
```

When EDITR is called, the tab character and tab stops are defaulted to a semicolon (;) and columns 7 and 21 even though they may have been redefined in a prior edit session and not changed back.

In a Multipoint environment, special considerations are necessary for setting tab stops. Consult CHAPTER 3 or the Multipoint Terminal Interface Subsystem User's Guide, HP Part No. 91730-90002, for additional information.

W (Set Window)

Allows user to limit the area of each record which is searched for a character string to be found, and/or exchanged.

Wa,b
a Initial column number of window (default is 1). b Final column number of window (default is 150).

EXAMPLE:

```
  /W7,9     <----The window is set to include only columns 7,8 and 9.
```

COMMENTS:

The set window command is especially useful when used in conjunction with the search commands (F, B, D, and J) and the exchange commands (G, Y, X, Z, V, and U). See the X command (for exchanging string patterns) for an example of using the set window command. Only the first character of a search pattern or exchange pattern is required to be within the window.

When EDITR is called, the display field or "window" consists of columns 1 through 150. This command resets the window to new boundaries.

(Sequence Numbers)

Allows user to place sequence numbers (in columns 76 through 80) on all lines in a file. Also, a three column identifier (in columns 73 through 75) can be specified by the command.

```
# [xxx [n1 [,n2 ]]]
```

xxx
Three character identifier. It occupies columns 73-75 and must be included when specifying n1 and n2 (it may contain blanks).

n1
Starting sequence number. If omitted, numbers start with 00000.

n2
Incrementing value for sequence numbers. n1 is incremented by n2 for each line. If omitted, n1 is incremented by 10.

EXAMPLES:

Given a file containing the following three lines:

```
  THIS IS LINE ONE OF SEQUENCING EXAMPLE
  THIS IS LINE TWO OF SEQUENCING EXAMPLE
  THIS IS LINE THREE OF SEQUENCING EXAMPLE
```

any one of the following four commands can be given (in all the examples it is assumed that line 0001 is the pending line). The resulting edited file is also shown.

```
1.  /#      <----Sequence lines with no three character identifier
                 and default values for n1 and n2.

  THIS IS LINE ONE OF SEQUENCING EXAMPLE                    00000
  THIS IS LINE TWO OF SEQUENCING EXAMPLE                    00010
  THIS IS LINE THREE OF SEQUENCING EXAMPLE                  00020
```

2. /#ABC <---Sequence lines with ABC as the three character identifier with default values for n1 and n2.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                    ABC00000
THIS IS LINE TWO OF SEQUENCING EXAMPLE                    ABC00010
THIS IS LINE THREE OF SEQUENCING EXAMPLE                  ABC00020
```

3. /#ABC,1 <---Sequence lines with ABC as the three character identifier, the first line starting with 00001 and n2 defaulted to 10.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                    ABC00001
THIS IS LINE TWO OF SEQUENCING EXAMPLE                    ABC00011
THIS IS LINE THREE OF SEQUENCING EXAMPLE                  ABC00021
```

4. /#ABC,1,1<---Sequence lines with ABC as the three character identifier, first line starting with 00001 and subsequent lines incremented by 1.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                    ABC00001
THIS IS LINE TWO OF SEQUENCING EXAMPLE                    ABC00002
THIS IS LINE THREE OF SEQUENCING EXAMPLE                  ABC00003
```

COMMENTS:

When listing a file on the terminal or the line printer, sequence numbers may be lost due to truncation. To delete the blanks in columns 60-70 so that the entire sequence number field can be output, these commands are entered at the terminal:

```
/W60,70        q-------Set window to delete blanks.
/Z           / <-------Exchange 10 blank spaces for no blank
                       spaces (see Z exchange command).
/3             <-------Number of lines to be changed (required
                       for Z command).
```

= (Set Line Length)

Allows user to reset the line length to another value and to truncate any characters in the line beyond the new limit.

=n
n Number of columns in line length.

EXAMPLE:

Given a file containing the following two lines:

```
  THIS IS LINE ONE OF THE LINE LENGTH EXAMPLE
  THIS IS LINE TWO OF THE LINE LENGTH EXAMPLE
```

The set line length command can be issued to truncate the lines to eliminate everything beyond column 8.

```
  /1    <----Makes line one the pending line.
  /=8   <----Changes line length to 8 columns.
```

The resulting file will look like:

```
    THIS IS
    THIS IS
```

If a third line was input at this point,

```
  / THIS IS LINE THREE OF THE LINE LENGTH EXAMPLE
```

The line would be truncated to include only the first 8 columns

```
  /P          <---Display pending line.
    THIS IS   <---Truncated third line.
```

COMMENTS:

The line length initially defaults to 150 characters when the EDITR is turned on unless a different line length is passed through the parameter string in the RU,EDITR command (see the CALLING EDITR section).

The line length command is useful for such things as eliminating line sequence numbers.

K (Kill Trailing Blanks)

Deletes trailing blanks from all lines in the work area.

K
No parameters required.

EXAMPLE:

Given a file with the following three lines of text:

```
  LINE ONE OF KILL EXAMPLE                                    00000
  LINE TWO OF KILL EXAMPLE                                    00010
  LINE THREE OF KILL EXAMPLE                                  00020
```

If the sequence numbers are eliminated via the set line length command,

```
  /=72    <---Columns 73 through 80 will be truncated.
```

the blanks up to column 72 will remain, thus requiring each record in in the file to be 72 characters.

```
  /H      <---Display number of characters in pending line.
    72    <---Number of characters in pending line.
```

To eliminate these unnecessary blanks from the file, the kill trailing blanks command can be used.

```
  /1      <---Make line one the pending line.
  /K      <---Kill trailing blanks in the file.
```

The resulting lines of text will then be stored more efficiently by making each record (or line of text) only as long as necessary.

```
/1      <---Make line one the pending line.
/P      <---Display pending line.

  LINE ONE OF KILL EXAMPLE

/H      <---Display number of characters in pending line.
  24    <---Number of characters in pending line (always even).
```

COMMENTS:

When a line of text is input through the EDITR, only the characters input prior to the carriage return are stored in a record. If the line

```
/ LINE ONE OF KILL EXAMPLE
```

had been input and immediately followed by a carriage return, there would only be 24 characters on that line. The kill trailing blanks command would not be necessary in this case.

The kill trailing blanks command always causes the source work area to be replaced by the destination work area, and always ends with an EOF message.

M (Merge Source File)

Allows user to merge a specified file after the pending line.

Mnamr
namr filename [:security code [:cartridge reference number]]

EXAMPLE:

This example will merge two files, a source file called FILE1 and a file to be merged called FILE2.

Following is a listing of the source file, FILE1:

```
  :LI,FILE1     <-----FMGR command to list FILE1.

FILE1  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001  LINE ONE OF SOURCE FILE.
0002  LINE TWO OF SOURCE FILE.
```

Following is a listing of the file to be merged, FILE2:

```
  :LI,FILE2     <-----FMGR command to list FILE2.

FILE2  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE OF FILE TO BE MERGED.
0002 LINE TWO OF FILE TO BE MERGED.
```

To merge FILE2 at the end of FILE1, the following commands should be input at the terminal.

```
  :RU,EDITR                       <---FMGR command to run EDITR.
  SOURCE FILE?                    <---EDITR response requesting file to
                                      be edited.
  /FILE1                          <---Name of file to be edited.
    LINE ONE OF SOURCE FILE.      <---First line of FILE1.
  //                              <---EDITR command to advance pending
                                      line by one line.
    LINE TWO OF SOURCE FILE.      <---New pending line.
  /MFILE2                         <---EDITR command to merge FILE2 after
                                      pending line.
  EOF
  /ER                             <---EDITR command to end edit session,
                                      replacing original version of FILE1
                                      with edited version.

  END OF EDIT
```

The edited version of FILE1 can now be listed to show the effect of the merge command.

```
  :LI,FILE1                       <---FMGR command to list FILE1.
FILE1 T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE OF SOURCE FILE.
0002 LINE TWO OF SOURCE FILE.
0003 LINE ONE OF FILE TO BE MERGED.
0004 LINE TWO OF FILE TO BE MERGED.
```

COMMENTS:

The file to be merged can be merged in after any desired point in the source file. The merge takes place after the pending line and before the next line. The next line becomes the new pending line.

L (Display a Number of Lines)

Displays a specified number of lines, starting with the pending line.

Ln [,lu]
n Number of lines to be printed (starting with pending line). lu Logical unit number of list device; default is user's terminal.

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

While on line one as the pending line, the first two lines can be displayed by the following command:

```
/L2              <---EDITR command to display two lines of text.
  LINE ONE.      <---First line of text displayed.
  LINE TWO.      <---Second line of text displayed.
```

COMMENTS:

When a file is listed on the line printer, using the File Manager LI command, qline numbers will precede each line of text. Using FILEA above as an example :

```
:LL,6            <---Change list device to line printer (LU 6).
:LI,FILEA        <---FMGR command to list file.
```

The display on the line printer will look like:

```
FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE.
0002 LINE TWO.
0003 LINE THREE.
```

If it desired that the file heading and line numbers not be listed, the EDITR "L" command can be used to list the file. In addition, the optional parameter, LU, can be specified to list the file on devices other than the default list device which is the user's terminal.

If only the pending line is desired to be displayed on he user's terminal, the EDITR P command may, also, be used. Using the above example to demonstrate:

```
/1              <---Make line one pending line and display it.
  LINE ONE.
/P              <---Display pending line.
  LINE ONE.
```

n (Display a Specified Line and Make it Pending Line)

Displays the requested line and makes it the pending line.

n
n The line number of the line to be displayed and made pending line.

EXAMPLE:

Given FILEA containing the following three lines of text:

```
  LINE ONE.
  LINE TWO.
  LINE THREE.
```

Assuming the pending line is the first line in the file and the third line is required to be displayed and will become the new pending line, the following command can be issued:

```
  /3              <---EDITR command to display third line of file
                      and make it pending line.
    LINE THREE.   <---Displayed third line of file.
  /P              <---Display pending line.
    LINE THREE.   <---Displayed pending line.
```

COMMENTS:

If the line number requested is less than or equal to the current pending line number, the destination work area replaces the source work area, changing text line numbers, (if any insertions or deletions were made to the text).

/ or + (Space Down a Number of Lines)

Advances the pending line the specified number and displays the new pending line.

+[n[,lu]] or /[n[,lu]]
n The number of lines to be skipped (default is one). lu Logical unit number of device on which the display occurs (default is the user's terminal). EDITR only recognizes this parameter when command is used in conjunction with an exchange command (G,Y, X,Z,V, and U).

EXAMPLE:

Given FILEA containing the following three lines,

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode and line one is the pending line, you can make line three the pending line and display it by using the space down a number of lines command.

```
/P            <--------------Display pending line.
  LINE ONE.   <--------------Pending line displayed.
/+2           <--------------Skip two lines and display pending
                             line.
  LINE THREE.<--------------Pending line displayed.
```

The thing can be done by using the slash (/) instead of the plus (+) sign.

```
/P              <--------------Display pending line.
  LINE ONE.     <--------------Pending line displayed.
//2             <--------------Skip two lines and display pending
                               line.
  LINE THREE.<--------------Pending line displayed.
```

COMMENTS:

The optional lu parameter can be used in conjunction with an exchange command. The following commands would be an example of this:

```
/1              <--------------Makes line one pending line.
  LINE ONE.     <--------------Pending line displayed.
/XLINE/LINE NUMBER<---------Exchange each occurrence of the word
                               LINE with LINE NUMBER.
/+3,6           <--------------Make the above requested exchange of
                               words for three lines starting from
                               the pending line and print the
                               changed lines on the printer which
                               is LU 6.
```

The changed lines will then be printed on the line printer. If the lu parameter had been omitted, the changed lines would have been displayed on the user's terminal.

N (Display a Pending Line Number)

Displays the line number of the pending line in the source work area.

N
No parameters required

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode and line one is the pending line then you could give the following commands at your terminal

```
/P            <--------------Display pending line.
  LINE ONE.   <--------------Pending line displayed.
/N            <--------------Display pending line number.
  1           <--------------Pending line number.
/+2           <--------------Space down two lines and display
                             pending line.
  LINE THREE.<--------------Pending line displayed.
/N            <--------------Display pending line number.
  3           <--------------Pending line number.
```

COMMENTS:

For an explanation of what the source work area is see the EDITR Work Area Section and Figure 1-1 in Chapter 1.

ND (Display Line Number in Destination Work Area)

Displays the line number of the last line written to the destination work area. The source work area pending line and EOF's are not included in the count.

ND
No parameters required

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode and line one is the pending line then you could input the following commands:

```
/P            <--------------Display pending line.
  LINE ONE.   <--------------Pending line displayed.
/ND           <--------------Display line number in destination
                             work area.
              <--------------Since no lines have been sent to
                             the destination work area yet, no
                             line number is displayed.
//            F--------------Space down one line and display
                             pending line.
  LINE TWO.   <--------------Displayed pending line.
/ND           <--------------Display line number in destination
                             work area.
  1           <--------------Line number of last line written to
                             the destination work area.
/+3           <--------------Space down three lines and display
                             pending line.
EOF           <--------------End of file.
```

```
/ND          <--------------Display line number in destination
                            work area.
  3          <--------------Line number of last line written to
                            the destination work area.
                            (Remember that the file contains
                            only three lines of text, all
                            three lines have been transferred
                            to the destination work area, and
                            the EOF is not included in the count)
```

COMMENTS:

For a description of the EDITR source and destination work areas see the EDITR Work Areas section in this Chapter and Figure 1-1 in Chapter 1.

H (Display Number of Characters in Pending Line)

Displays the number of characters in the pending line. The count includes a padded blank if the number of characters is uneven.

H
No parameters required

EXAMPLE:

FILEA contains the following two lines of text:

```
   THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
   THIS LINE HAS FORTY-SIX ASCII CHARACTERS.
```

If you are in the edit mode then you could input the following commands at your terminal

```
   /P            <--------------Display pending line.
     THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
   /H            <--------------Display number of characters in
                                pending line.
     42          <--------------Number of characters in pending line
                                (includes padded blank).
   //            <--------------Space down one line and display
                                pending line.
     THIS LINE HAS FORTY-SIX ASCII CHARACTERS.
   /H            <--------------Display number of characters in
                                pending line.
     46          <--------------Number of characters in pending line.
```

COMMENTS:
The count includes a padded blank if the number of characters is uneven, because EDITR generates word-length records of two characters per word. This padding may be lost during character edits but replaced as the line is added to the destination work area as shown in the following example.

```
:RU,EDITR     <--------------FMGR Command to run EDITR.
SOURCE FILE?  <--------------EDITR requesting name of file to be
                             edited.
/O            <--------------Put empty file into EDITR's source
                             work area.
```

The following command enters a line of text into the EDITR's source work area

```
/ THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
```

The line has not been transferred to the EDITR's destination work area yet; therefore, if you request the number of characters in the pending line the padded blank will not be included.

```
/H            <--------------Display number of characters in
                             pending line.
  41          <--------------Displayed number of characters.
```

Now, by moving the line to the EDITR's destination work area, the padded blank will be included when the number of characters is requested.

```
/1            <--------------Display first line of text and
                             make it the pending line.
  THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
/H            <--------------Display number of characters in
                             pending line.
  42          <--------------Displayed number of characters.
```

HL (Display Header)

Displays a header which facilitates column location.

HL
No parameters required

EXAMPLE:

```
/HL             <--------------Display column header.
  ''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''/''''6''''/''''7''''
'8
```

^ (Back up in Destination Work Area)

Allows user to back up a specified number of lines in the destination work area and display the new pending line.

^[n]
n Number of lines to be backed up in destination work area. Default is to back up one line.

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, the following commands can be input:

```
/P             <--------------Display pending line.
  LINE ONE.    <--------------Pending line displayed.
/+2            <--------------Advance pending line two lines and
                              display new pending line.
  LINE THREE.<--------------Pending line displayed.
/^2            <--------------Back up pending line two lines and
                              display new pending line.
  LINE ONE.    <--------------Pending line displayed.
```

COMMENTS:

Following input of this command, EDITR copies the remainder of the source work area (pending line to end of file) into the destination work area. The destination work area then becomes the new source work area with the pending line moved back n lines. All lines prior to the new pending line are then copied into a new destination work area (lines 1 through n-1).

If n is so large that it causes the pointer to back up past the beginning of the destination work area, the error message ?? is displayed and a smaller value should be specified.

S (Display Approximate Number of Words in Destination Work Area)

Prints the approximate number of words in the destination work area.

S
No parameters required

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode you can input the following commands

```
/P              <--------------Display pending line.
  LINE ONE.     <--------------Pending line displayed.
/L3             <--------------Display three lines of text.  Command
  LINE ONE.                    also moves the three lines to the
  LINE TWO.                    destination work area by moving the
  LINE THREE.                  pending line down three lines.
EOF             <--------------End of file.
/S              <--------------Display number of words in destination
                               work area.
  19            <--------------Number of words in destination work
                               area.
```

COMMENTS:

This command can be used to determine the need to break up files into smaller segments which would result in large paper tapes. For example, a paper tape containing more than 14,000 words exceeds the standard box size. If the word count is determined with the S command, the file can be broken with a series of zero-length records in the appropriate places.

P (Edit and Display the Pending Line)

Edits and displays the pending line.

P[editstring]
editstring The editstring can be composed of delimiters used as place holders to preserve existing text, new text to be inserted or to replace existing text, and a non-printing control command to replace, insert or delete characters. If no editstring is specified, then the existing pending line is displayed. (Delimiters must be the current prompt character; default prompt character is a slash).

EXAMPLE:

FILEA contains the following two lines of text:

```
   LINE ONE.
   LINE TWO.
```

The following edit session demonstrates use of the "P" command.

```
   :RU,EDITR       <----------FMGR Command to run EDITR.
   SOURCE FILE?
   /FILEA          <----------File to be edited.
     LINE ONE.     <----------EDITR responds with first line of file.
   /P              <----------Display pending line.
     LINE ONE.     <----------Pending line displayed.
   /P/////FIRST.   <----------Edit pending line and display new
                               pending line.
     LINE FIRST.   <----------EDITR responds with new pending line.
   /ER             <----------End edit session replacing old
                               version of FILEA with edited version.
   END OF EDIT     <----------EDITR response indicating edit session
                               is over.
```

COMMENTS:

The O and C commands are similar to the P command when used for character edits (see the following two sections for a description of these commands).

The P command is also used frequently to display the unedited pending line. Many of the examples for other commands make use of this feature.

C (Edit Pending Line and Advance Line)

Allows editing of the pending line and advances the pending line following the edit.

C[editstring]
editstring The editstring can be composed of delimiters used as place holders to preserve existing text, new text to be inserted or to replace existing text, and a non-printing control command to replace, insert or delete characters. If no editstring is specified, then the pending line will be displayed, no change will be made to it, and the pending line will be advanced one line and displayed. (Delimiters must be the current prompt character; default prompt character is a slash).

EXAMPLE:

FILEA contains the following two lines of text:

```
LINE ONE
LINE TWO
```

To change the above two lines, the following set of commands can be input while in the edit mode:

```
/P              <----------Display pending line.
  LINE ONE      <----------Pending line displayed.
/C/////FIRST    <----------Edit pending line changing ONE to FIRST.
                           Display edited line, advance pending
                           line, and display new pending line.
  LINE FIRST    <----------Edited line displayed.
  LINE TWO      <----------New pending line.
/C/////SECOND   <----------Edit pending line changing TWO to
                           SECOND. Display edited line, advance
                           pending line, and display new pending
                           line.
  LINE SECOND   <----------Edited line displayed.
EOF             <----------End of file.
```

O (Duplicate Pending Line and Edit)

Duplicates the pending line and allows character edits on the duplicate.

O[editstring]
editstring The editstring is composed of delimiters used as place holders to preserve existing text, new text to be inserted or to replace existing text, and a non-printing control command to replace, insert or delete characters. If no editstring is specified, then the existing pending line is duplicated and displayed. (Delimiters must be the current prompt character; default prompt character is a slash).

EXAMPLE:

FILEA contains the single line of text:

```
   LINE ONE
```

If you are in the edit mode, the following EDITR commands can be input at your terminal to create two additional lines:

```
   /O              <-------------Duplicate pending line and display
                                 duplicate.
     LINE ONE      <-------------Duplicate line displayed.
   /P/////TWO      <-------------Change ONE to TWO and display edited
                                 line.
     LINE TWO      <-------------Edited line displayed.
   /O              <-------------Duplicate pending line and display
                                 duplicate.
     LINE TWO      <-------------Duplicate line displayed.
   /P/////THREE    <-------------Change TWO to THREE and display edited
     LINE THREE                  line.
   /1              <-------------Make line one pending line and display
                                 it.
```

```
  LINE ONE     <-------------Pending line displayed.
/L3            <-------------Display three lines of text.
  LINE ONE
  LINE TWO
  LINE THREE
EOF            <-------------End of file.
```

COMMENTS:

A more efficient method of accomplishing the above example would be to use the editstring parameter in the O command. For example:

```
/P             <-------------Display pending line.
  LINE ONE     <-------------Pending line displayed.
/O/////TWO     <-------------Duplicates pending line and
                             edits duplicate.
  LINE TWO           <-------Edited line displayed and made new
                             pending line.
/O/////THREE    <------------Duplicates pending line and
                             edits duplicate.
  LINE THREE    <------------Edited line displayed and made
                             new pending line.
```

R (Replace Pending Line with Text)

Replaces the pending line with new text entered at the terminal.

Rtext
text New line of text to replace pending line. If no new text is specified, the pending line becomes zero length.

EXAMPLE:

The following example demonstrates use of this command to replace the pending line with a new line of text.

```
/P              <--------------Display pending line.
  LINE ONE.     <--------------Pending line displayed.
/RFIRST LINE.<--------------Replace pending line text with new
                              text.
/P              <--------------Display pending line.
  FIRST LINE.<--------------Pending line displayed.
```

I (Insert Text Before Pending Line)

Inserts a new line of text before the pending line.

Itext
text Line of text to be inserted before pending line in destination work area. If command is entered with no new text, the inserted line becomes zero length.

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, you can input the following commands:

```
/P            <--------------Display pending line.
  LINE ONE.   <--------------Pending line displayed.
//            <--------------Advance pending line one line and
                             display new pending line.
  LINE TWO.   <--------------Pending line displayed.
/INEW LINE.   <--------------Insert new line of text before
                             pending line.
/1            <--------------Make line one pending line and
                             display pending line.
  LINE ONE.   <--------------Pending line displayed.
/L4           <--------------Display four lines of text.
  LINE ONE.
  NEW LINE.
  LINE TWO.
  LINE THREE.
EOF
```

{ } (Insert Text after Pending Line)

Used to insert new text immediately after the pending line. The new line then becomes the pending line. Note that {} represents a space.

{ }text
text New line of text to be inserted following pending line. If command is entered without new text, the new line has zero length.

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, then you could input the following commands:

```
/P            <--------------Display pending line.
  LINE ONE.   <--------------Pending line displayed.
/ NEW LINE.   <--------------Insert new line of text following
                             pending line and make it new pending
                             line.
/P            <--------------Display pending line.
  NEW LINE.   <--------------Pending line displayed.
/1            <--------------Make line one pending line and
                             display pending line.
  LINE ONE.   <--------------Pending line displayed.
/L4           <--------------Display four lines of text starting
                             from pending line.
  LINE ONE.
  NEW LINE.
  LINE TWO.
  LINE THREE.
EOF           <--------------End of file.
```

– (Delete a Number of Lines)

Deletes a specified number of lines in the text.

-n
n Number of lines of text to be deleted (starting with pending line). If no value is specified, one line is deleted.

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, then you could give the following set of commands:

```
/P            <--------------Display pending line.
  LINE ONE.   <--------------Pending line displayed.
/-2           <--------------Delete two lines of text and display
                             new pending line.
  LINE THREE.<--------------Pending line displayed.
/1            <--------------Make line one pending line and display
                             pending line.
  LINE THREE.<--------------Pending line displayed.
/L3           <--------------Display three lines of text.
  LINE THREE.
EOF           <--------------End of file.
  LINE THREE.
EOF           <--------------End of file.
```

CNTL/R (Replace Characters)

Used in character edits to replace characters on a line.

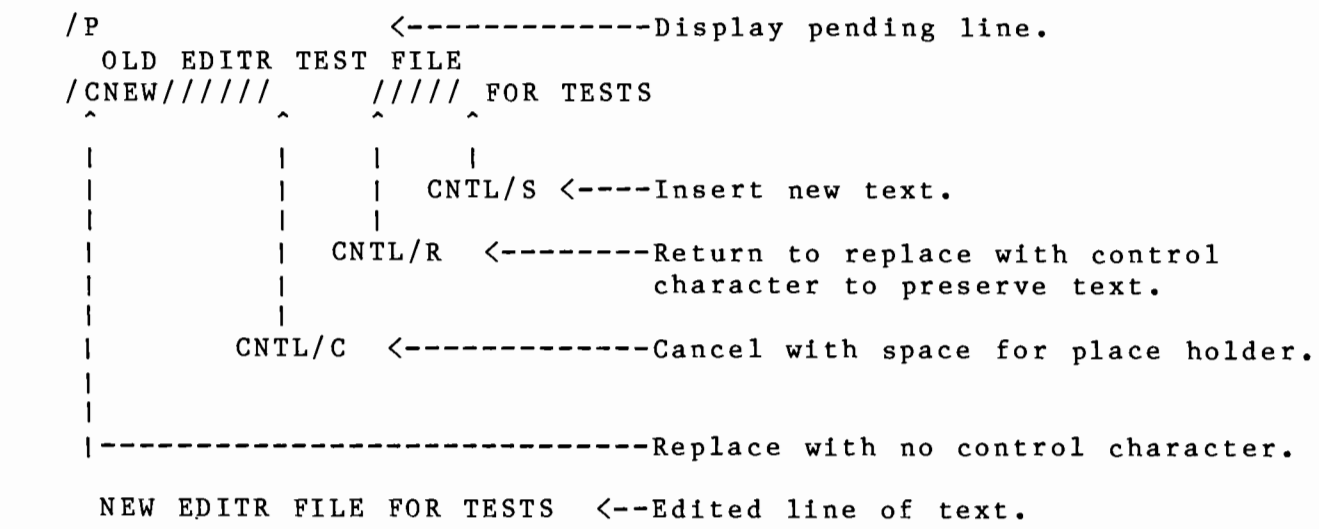
CNTL/R
A control R is input by striking the "R" key while pressing the "CNTL" key on the terminal. It is a non-printing character. Refer to EDITR In a Multipoint Environment, Chap 3. ;CNTL and ESC commands are inoperative in a Multipoint Environment.

COMMENTS:

Character replacement is the EDITR default mode so that the control character (CNTL/R) can be omitted when EDITR is initially turned on, or following a P,C, or O command. It is only required following one of the control characters CNTL/I, CNTL/S, or CNTL/C.

Any embedded character you wish to preserve in the original text is skipped by entering the current delimiter (/). Using the tab character will also cause characters to be skipped. Skipped characters in the new line appear exactly as they did in the old line. A carriage return preserves the rest of the line unless CNTL/T was specified to truncate.

An example of a character edit combination where the CNTL/R is required is shown below.



CNTL/I or CNTL/S (Insert Characters)

Used to insert characters in the pending line.

CNTL/I or CNTL/S
A control I or S is input by striking the "I" or "S" key while pressing the "CNTL" key on the terminal. They are non-printing characters. Refer to EDITR In a Multipoint Environemnt in Chapter 3. CNTL and ESC are inoperative in a Multipoint Environment.

EXAMPLES:

1) CNTL/S example.

```
/P              <---------------Display pending line.
  LINE ONE.     <---------------Pending line displayed.
/P//////// OF TEXT <-----------Edit pending line and display it.
          ^
          |
          -- CNTL/S

  LINE ONE OF TEXT.   <--------Edited version of pending line.
```

2) CNTL/I example.

```
/P              <---------------Display pending line.
  LINE ONE.     <---------------Pending line displayed.
/P////////          <-----------Edit pending line and display it.
          ^
          |
          -- CNTL/I
 OF TEXT            <-----------CNTL/I also acts as a carriage return.

 LINE ONE OF TEXT. <-----------Edited version of pending line.
```

COMMENTS:

The CNTL/I and CNTL/S commands are used with the P,C, and O commands (for a description of these commands, see the explanations given earlier in this Chapter).

Once CNTL/I or CNTL/S is entered, entering the current delimiter (/), to do character skipping, inserts blanks. The tab character also causes the tabbed number of blanks to be inserted.

CNTL/C (Cancel Characters)

Used to delete characters from the pending line.

CNTL/C
A control C is input by striking the "C" key while pressing the "CNTL" key on the terminal. It is a non-printing character. Refer to the EDITR In a Multipoint Envitonment in Chapter 3. CNTL and ESC are inoperative in a Multipoint Environment.

EXAMPLE:

```
/P                      <---------Display pending line.
  LINE ONE OF TEXT.     <---------Pending line displayed.
/P////////XXXXXXXX      <---------Edit pending line and display.
          ^
          |
        CNTL/C    <-------------Delete characters.

  LINE ONE.             <---------Edited version of pending line.
```

COMMENTS:

Each character or place holder entered after CNTL/C will delete one character from the pending line. Character skipping, i.e., entering the current delimiter deletes characters. The tab character deletes everything up to the tab stop. When carriage return is entered, the pending line will be left-justified.

The CNTL/C command is used in conjunction with the P,C, or O commands. For a description of these commands, see the explanations given earlier in this Chapter.

CNTL/T (Truncate Characters)

Used to delete characters from the end of the pending line. When the control character is entered, the remainder of the line is eliminated.

CNTL/T
A control T is input by striking the "T" key while pressing the "CNTL" key on the terminal. It is a non-printing character. Refer to EDITR in a Multipoint Environment in Chapter 3. CNTL and ESC commands are inoperative in a Multipoint Environment.

EXAMPLE:

```
/P                <-------------Display pending line.
  LINE ONE OF TEXT  <-----------Pending line displayed.
/P////////          <-------------Edit pending line and display.
          ^
          |
       CNTL/T      <-------------Eliminate characters following CNTL/T.

  LINE ONE         <-------------Pending line displayed.
```

COMMENTS:

When operating under multipoint, the CNTL/T command can be replaced by the Q command. For details on the Q command see CHAPTER 3 for EDITR In a Multipoint Environment.

B (Find a Line with a Find Field — SOF to EOF)

Searches the source work area, from the start-of-file to the end-of-file, for the first line of text which matches the find field. When a line containing the correct character string is found, it becomes the new pending line. The search ends at EOF if no match is found.

```
B                <------------Search for zero length record.
 ^
 |---CNTL/@

Btext            <------------Search for left justified text.

Btext            <------------Search for text anywhere in line.
 ^
 |---ESC key

B/text           <------------Search for text in window.

B;text           <------------Search for field at tab stop.

B                <------------Successive search for same field.
                              The same line will always be found
                              by this command.
```

CNTL/@
A control @ is input by striking the "@" key while pressing the "CNTL" key on the terminal. It is a non-printing character.

text
This is the portion of text that is to be found by the B command.

ESC
The ESC key is located on the terminal. It is a non-printing character.

```
EXAMPLES:

FILEA contains the following four lines of text:f

   LINE ONE OF EXAMPLE.
   LINE TWO OF EXAMPLE.
                                    <---(zero length record)
         LINE FOUR OF EXAMPLE.

1) B CNTL/@ example.

   /B              <---------------Search for zero length record and
     ^                              make it pending line.
     |
     ---CNTL/@
                   <---------------Pending line (zero length record).
   /^              <---------------Move pending line back one line and
                                    display new pending line.
     LINE TWO OF EXAMPLE.  <------Pending line displayed.

2) Btext example.

   /BLINE          <---------------Search for first occurrence of LINE
                                    (left justified), making line
                                    pending line, and display it.
     LINE ONE OF EXAMPLE.  <------Pending line displayed.

3) Btext example.
    ^
    |--ESC

   /BTWO           <---------------Search for first occurrence of TWO,
     ^                              making line containing TWO pending
     |--ESC                        line, and display it.
     LINE TWO OF EXAMPLE.  <------Pending line displayed.

4) B/text example.

   /W6,11          <---------------Make window between columns 6
                                    through 11.
   /BLINE          <---------------Find first occurrence of LINE within
                                    window constraints, making line
     ^                              containing text pending line,
     |--ESC                        and display pending line.
            LINE FOUR OF EXAMPLE.
```

5) B;text example.

```
/B;LINE          <----------------Find first occurrence of LINE at tab
                                  stop, making it pending line and
                                  display pending line.
          LINE FOUR OF EXAMPLE.
```

6) B example.

```
/B;LINE          <----------------Find first occurrence of LINE at tab
                                  stop, making it pending line and
                                  display pending line.
          LINE FOUR OF EXAMPLE.
/1               <----------------Make line one pending line and
                                  display pending line.
  LINE ONE OF EXAMPLE.
/B               <----------------Search for same field sought
                                  previously.
          LINE FOUR OF EXAMPLE.
```

COMMENTS:

The B command always starts at the first line of the file to find the text. Once the text is found, the line containing it becomes the pending line. The B command cannot be used for successive searches for the same text, because it will always find the same line unless the text being searched for is changed on that line. All lines passed over in the search are written to the destination work area; they are not deleted. The B command always forces the destination work area to replace the source work area before the search.

The slash in the command format B/text, is representative of the initial delimiter. If the delimiter has been changed (X - CHANGE EDITR PROMPT CHARACTER), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (T - SET TAB STOPS), the new tab character must be used in place of the ;.

Note that if the window is set at its default values (1 to 150), the B/text command does the same thing as the B^<ESC>text.

F (Find a Line with Find Field from Pending Line to EOF)

Causes a search of the source work area, beginning at the line following the pending line and continuing to the end-of-file for a line containing a specified character string. When a line containing this field is found, it becomes the new pending line. If the line is not found, the search ends at the end-of-file.

```
Ftext          <-------------Search for left justified field.

Ftext          <-------------Search for field embedded anywhere in
 ^                            line.
 |--ESC

F/text         <-------------Search for field within a window.

F;text         <-------------Search for field beginning at tab
                             stop.

F              <-------------Successive searches for same field.
```

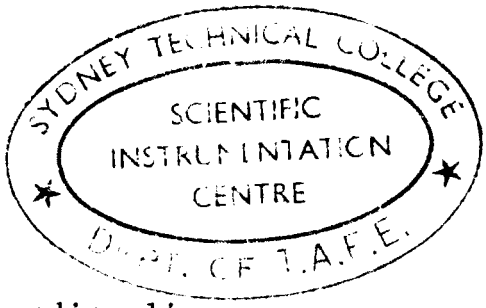
```
text
This is the portion of text that is to be found by the F command.

ESC
The ESC key is located on the terminal.  It is a non-printing
character.
```

EXAMPLE:

FILEA contains the following four lines of text:

```
   THIS IS AN EXAMPLE
   FOR DEMONSTRATING
   THE F COMMAND
          THE F COMMAND
```



1) Ftext example.

```
/P                      <--------Display pending line.
  THIS IS AN EXAMPLE    <--------Pending line displayed.
/FTH                    <--------Search for left justified field
                                 with text TH, making line containing
                                 field pending line, and display it.
  THIS IS AN EXAMPLE    <--------Pending line displayed.
/+                      <--------Advance pending line one line and
                                 display new pending line.
  FOR DEMONSTRATING     <--------New pending line displayed.
/FTH                    <--------Search for left justified field
                                 with text TH, making line containing
                                 field pending line, and display it.
  THE F COMMAND         <--------Pending line displayed.
```

2) Ftext example.

```
  ^
  |--ESC

/P                      <--------Display pending line.
  THIS IS AN EXAMPLE    <--------Pending line displayed.
/FDEMO                  <--------Search for text embedded anywhere
  ^                              in line, making line pending line,
  |--ESC                         and display it.

  FOR DEMONSTRATING     <--------Pending line containing text.
/+                      <--------Advance pending line by one line
                                 and display new pending line.
  THE F COMMAND         <--------Pending line displayed.
/FDEMO                  <--------Search for text embedded anywhere
  ^                              in line, making line pending line,
  |--ESC                         and display it.

EOF                     <--------End of file.  Indicates remainder of
                                 of file was searched and text was
                                 not found.
```

3) F/text example.

```
/P                      <--------Display pending line.
  THIS IS AN EXAMPLE    <--------Pending line displayed.
/W6,10                  <--------Set windown between columns 6 and 10.
/F/COM                  <--------Search for text within window, making
                                 line containing text pending line,
                                 and display pending line.
  THE F COMMAND         <--------Pending line displayed.
```

4) F;text example.

```
/P                      <--------Display pending line.
  THIS IS AN EXAMPLE    <--------Pending line displayed.
/F;THE                  <--------Search for text beginning at tab
                                 stop, making line containing text
                                 pending line, and display pending
                                 line.
          THE F COMMAND   <-----Pending line displayed.
```

5) F example.

```
/P                      <--------Display pending line.
  THIS IS AN EXAMPLE    <--------Pending line displayed.
/FTHE                   <--------Search for text embedded anywhere
  ^                              in line, making line containing
  |--ESC                         text new pending line, and display
                                new pending line.

  THE F COMMAND         <--------Pending line displayed.
/F                      <--------Successive search for same text,
                                 making line containing text new
                                 pending line, and display line.
          THE F COMMAND   <-----Pending Line displayed.
```

COMMENTS:

The slash in the command format F/text, is representative of the initial delimiter. If the delimiter has been changed (see the X CHANGE EDITR PROMPT CHARACTER section), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (see the SET TAB STOPS section), the new tab character must be used in place of the ;.

Note that if the window is set at its default values (1 to 150), then the F/text command does the same thing as the F^<ESC> text.

D (Delete Lines to Find Field or EOF)

Deletes a block of text from the pending line to the line containing a specified field. After the deletion, the line of text containing the specified field becomes the new pending line. If the specified field is not encountered, the remainder of the file is deleted. If this command is used without specifying a field, the last field entered is used.

```
D                <-------------Delete lines until a zero length field
 ^                               is encountered.
 |--CNTL/@

Dtext            <-------------Delete lines until specified text is
                                 found in left justified field.

Dtext            <-------------Delete lines until specified text is
 ^                               found in field located anywhere in
 |--ESC                         line.

D/text           <-------------Delete lines until specified text is
                                 found in field located within window.

D;text           <-------------Delete lines until specified text is
                                 found in field beginning at tab stop.

D                <-------------Successive deletes to specified text
                                 in prior delete command.
```

PARAMETERS:

CNTL/@
A control @ is input by striking the "@" key while pressing the "CNTL" key on the terminal. It is a non-printing character.

text
This is the portion of text that is to be found by the D command.

ESC
The ESC key is located on the terminal. It is a non-printing character.

EXAMPLE:

FILEA contains the following 7 lines of text

```
   THIS IS AN EXAMPLE
   DEMONSTRATING THE
                          <--------Zero length record.
   DELETE LINES COMMAND.
                          <--------Zero length record.
   THIS NEXT LINE BEGINS
         IN COLUMN 7.

1) D   example.
    ^
    |--CNTL/@

   /1                <------------Make line one pending line and display
                                  it.
     THIS IS AN EXAMPLE  <--------Pending line displayed.
   /D                <------------Delete all lines until zero length
     ^                            record is found.  Make zero length
     |--CNTL/@                    record pending line and display it.

                     <------------Zero length record displayed.
   /1                <------------Make line one pending line and display
                                  it.
                     <------------Pending line displayed (zero length
                                  record).
   /L6               <------------Display six lines of text.

     DELETE LINES COMMAND.

     THIS NEXT LINE BEGINS
           IN COLUMN 7.
   EOF               <------------End of file.
```

2) Dtext example.

```
/1                  <------------Make line one pending line and display
                                 it.
  THIS IS AN EXAMPLE  <--------Pending line displayed.
/DTHIS              <------------Delete all lines until THIS is
                                 encountered in a left justified
                                 field, making line containing
                                 text pending line, and displaying it.
  THIS NEXT LINE BEGINS  <-----Pending line displayed.
/1                  <------------Make line one pending line and display
                                 it.
  THIS NEXT LINE BEGINS  <-----Pending line displayed.
/L6                 <------------Display six lines of text.
  THIS NEXT LINE BEGINS
        IN COLUMN 7.
EOF                 <------------End of file.
```

3) Dtext example.

```
 ^
 |--ESC

/1                  <------------Make line one pending line and display
                                 it.
  THIS IS AN EXAMPLE     <-----Pending line displayed.
/DBEGINS            <------------Delete all lines until BEGINS is
  ^                              encountered anywhere in file, making
  |--ESC                         line containing BEGINS pending line,
                                 and displaying it.
  THIS NEXT LINE BEGINS  <-----Pending line displayed.
/1                  <------------Make line one pending line and display
                                 it.
  THIS NEXT LINE BEGINS  <-----Pending line displayed.
/L6
  THIS NEXT LINE BEGINS
        IN COLUMN 7.
EOF                 <------------End of file.
```

4) D/text example.

```
/1                  <------------Make line one pending line and display
                                 it.
  THIS IS AN EXAMPLE     <-----Pending line displayed.
/W10,15             <------------Set window between columns 10 and 15.
/D/LINE             <------------Delete all lines until LINE is located
                                 in a field within the specified
                                 window.  Make line containing LINE
                                 pending line and display it.
```

```
  THIS NEXT LINE BEGINS  <-----Pending line displayed.
/1                       <------------Make line one pending line and display
                                      it.
  THIS NEXT LINE BEGINS  <-----Pending line displayed.
/L6                      <------------Display six lines of text.
  THIS NEXT LINE BEGINS
        IN COLUMN 7.
EOF                      <------------End of file.
```

5) D;text example.

```
/1                       <------------Make line one pending line and display
                                      it.
  THIS IS AN EXAMPLE     <-----Pending line displayed.
/D;IN                    <------------Delete all lines until IN is
                                      encountered at the first tab stop
                                      on some line.  Make line containing
                                      IN pending line and display it.
        IN COLUMN 7.     <-----Pending line displayed.
/1                       <------------Make line one pending line and display
                                      it.
        IN COLUMN 7.     <-----Pending line displayed.
/L6                      <------------Display six lines of text.
        IN COLUMN 7.
EOF                      <------------End of file.
```

6) D example.

```
/1                       <------------Make line one pending line and display
                                      it.
  THIS IS AN EXAMPLE     <-----Pending line displayed.
/D                       <------------Delete all lines until a zero length
  ^                                   record is encountered, making zero
  |--CNTL/@                           length record pending line and display
                                      it.
                         <------------Zero length record displayed as
                                      pending line.
/L6                      <------------Display six lines of text.
```

DELETE LINES COMMAND

```
  THIS NEXT LINE BEGINS
       IN COLUMN 7.
EOF              <------------End of file.
/1               <------------Make line one pending line and display
                              it.
                       <-----Pending line displayed (zero length
                              record).
/D               <------------Delete all lines until text field
                              specified in last delete command is
                              encountered.  Make line containing
                              text new pending line and display it.
                       <-----Pending line displayed (zero length
                              record).
 /L6              <------------Display six lines of text.

  THIS NEXT LINE BEGINS
       IN COLUMN 7.
EOF              <------------End of file.
```

COMMENTS:

The slash in the command for D/text, is representative of the initial delimiter. If the delimiter has been changed (see the X CHANGE EDITR PROMPT CHARACTER section), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (see the SET TAB STOPS section), the new tab character must be used in place of the ;.

Note that if the window is set at its default values (1 to 150), then the

```
D/text
```

command does the same thing as the

```
D^<ESC>text.
```

J (Jump to Find Field Line and make it it Now Pending Line)

Used to either delete or move text. It causes the pending line pointer to jump to a line containing a specified text string in the source work area without moving the jumped over lines to the destination work area.

```
Jtext            <------------Jump to line containing text in left
                              justified field.

Jtext            <------------Jump to line containing text embedded
 ^                            anywhere in line.
 |--ESC

J/text           <------------Jump to line containing text embedded
                              anywhere within window.

J                <------------Jump to same line that was jumped to
                              with prior jump command.
```

```
text
This is the portion of text that is to be found by the J command.

ESC
The ESC key is located on the terminal.  It is a non-printing
character.
```

EXAMPLE:

```
Source Work Area      Pending Line      Destination Work Area
----------------      ------------      ---------------------
FIRST LINE            <----PL
      SECOND LINE
THIRD LINE
      FOURTH LINE

/JTHIRD     <--------------------Jump to line with text in left
                                justified field.
```

```
Source Work Area      Pending Line      Destination Work Area
----------------      ------------      ---------------------
FIRST LINE                              FIRST LINE
      SECOND LINE
THIRD LINE            <----PL
      FOURTH LINE

/JCOND      <--------------------Jump to line with text embedded
  ^                              in line.
  |--ESC

Source Work Area      Pending Line      Destination Work Area
----------------      ------------      ---------------------
FIRST LINE                              FIRST LINE
      SECOND LINE     <----PL           THIRD LINE
THIRD LINE
      FOURTH LINE

/W10,20     <--------------------Set window between columns 10 to 20.
/J/LINE     <--------------------Jump to line containing first
                                 occurrence of text which is embedded
                                 within window.

Source Work Area      Pending Line      Destination Work Area
----------------      ------------      ---------------------
FIRST LINE                              FIRST LINE
      SECOND LINE     <----PL           THIRD LINE
THIRD LINE                                    SECOND LINE
      FOURTH LINE

/J          <--------------------Jump to same line as previous jump
                                 command.

Source Work Area      Pending Line      Destination Work Area
----------------      ------------      ---------------------
FIRST LINE                              FIRST LINE
      SECOND LINE     <----PL           THIRD LINE
THIRD LINE                                    SECOND LINE
      FOURTH LINE                             SECOND LINE

/1          <--------------------Make line one pending line.  Also,
                                 copies remainder of source work area
                                 from pending line on down into
                                 destination work area and then makes
                                 destination work area new source work
                                 area.
```

```
Source Work Area      Pending Line      Destination Work Area
----------------      ------------      ---------------------
FIRST LINE            <----PL
THIRD LINE
      SECOND LINE
      SECOND LINE
      SECOND LINE
THIRD LINE
      FOURTH LINE
```

COMMENTS:

Before using the J command for deleting and moving blocks of text, you should be familiar with the concept of source and destination work areas (refer to the EDITR WORK AREAS). Remember that whenever the pending line pointer goes back to a prior line in the source work area, the destination work area will become the new source work area.

Whenever text is sought with a J command, the first line containing the text in the source work area will become the pending line regardless of where the pending line pointer was prior to the command. The only line moved by the J command to the destination work area is the pending line prior to the jump.

The slash in the command format J/text, is representative of the initial delimiter. If the delimiter has been changed (see the CHANGE EDITR PROMPT CHARACTER section), the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (see the SET TAB STOPS section), the new tab character must be used in place of the ;.

Note that if the window default values (1 to 150) are used, the

```
J/text
```

command does the same thing as the

```
J^<ESC>text.
```

G (Character Replace on Pending Line)

Exchanges an old character string with a new character string on the pending line. The new pending line following the exchange is then displayed.

Goldstring/newstring
oldstring Old character string to be exchanged with new. Can be located anywhere in pending line. If several identical character strings are located on the line, all of them will be exchanged for the new string. Can be any length except zero. newstring This is the new character string which is to replace the old string. Can be any length including zero.

EXAMPLE:

```
/P                <----------------Display pending line.
  OLD STRING              <-------Pending line displayed.
/GOLD/NEW         <----------------Exchange OLD text for NEW text and
                                   display new pending line.
  NEW STRING              <-------Pending line displayed.
```

COMMENTS:

If the pending line contains several identical strings of text, but not all of them are desired to be changed, then the set window command can be used to selectively change character strings. For example, if the pending line is

```
OLD OLD OLD OLD
```

and only the third occurrence of OLD is desired to be exchanged for NEW, the following commands can be input to accomplish this.

```
/W8,12            <----------------Set window between columns 8 and 12.
/GOLD/NEW         <----------------Exchange OLD text with NEW text and
```

```
                                    display new pending line.
OLD OLD NEW OLD          <--------Pending line displayed.
```

Once the Goldstring/newstring Command has been issued, subsequent exchanges may be made by specifying G without any parameters, i.e., G parameters remain in effect until changed.

The exchange fields used in the G command are not tabbed, so that the tab character can be used like any other character in the exchange string.

The only time the current delimiter is recognized as a delimiter is the first time it occurs separating the old string and the new string.

Y (Exchange on Pending Line, Display next Occurrence of Pattern)

Performs an exchange of an old string with a new string on the pending line.

Yoldstring/newstring <------	Exchange all occurrences of old string on pending line with new string. After exchange, the next occurrence of old string becomes the new pending line. If same exchange on new pending line is desired, only the command Y need be input.

oldstring
Old string of text to be exchanged with new text string. It can be any length except zero.

newstring
This is the new text string which is to replace the old. It can be any length including zero.

EXAMPLE:

FILEA contains the following five lines of text:

```
THIS IS AN OLD STRING.
THE OLD STRING CAN BE
REPLACED BY A NEW
STRING WITH THE Y
COMMAND.
```

To change all occurrences of the text, STRING, with the text, TEXT, the following commands can be input while in the edit mode.

```
/YSTRING/TEXT      <-----------Exchange STRING with TEXT on pending
                               line, display edited line, find next
                               occurrence of STRING, making it the
                               pending line and display it.
```

```
  THIS IS AN OLD TEXT.    <-----Edited line displayed.
  THE OLD STRING CAN BE   <-----New pending line displayed.
/Y            <-----------------Exchange STRING with TEXT on pending
                                line, display edited line, find next
                                occurrence of STRING, making it
                                pending line and display it.
  THE OLD TEXT CAN BE     <-----Edited line displayed.
  STRING WITH THE Y       <-----New pending line displayed.
/Y            <-----------------Exchange STRING with TEXT on pending
                                line, display edited line, find next
                                occurrence of STRING, making it
                                pending line and display it.
  TEXT WITH THE Y         <-----Edited line displayed.
EOF           <-----------------End of file.
```

COMMENTS:

A combination of Y and F commands can be used to do selective exchanges. The F command is entered (without parameters) to skip an exchange on this occurrence and find the next occurrence. Using FILEA as an example, the first and last occurrence of STRING can be changed to TEXT leaving the second occurrence unchanged by the following commands

```
/YSTRING/TEXT     <-------------Exchange STRING with TEXT on pending
                                line, display edited line, find next
                                occurrence of STRING, making it the
                                pending line and display it.
  THIS IS AN OLD TEXT.    <-----Edited line displayed.
  THE OLD STRING CAN BE   <-----New pending line displayed.
/F                <-------------Find next occurrence of STRING and
                                make it pending line leaving present
                                pending line unchanged.
  STRING WITH THE Y       <-----New pending line displayed.
/Y            <-----------------Exchange STRING with TEXT on pending
                                line, display edited line, find next
                                occurrence of STRING, making it
                                pending line and display it.
  TEXT WITH THE Y         <-----Edited line displayed.
EOF           <-----------------End of file.
```

If the string to be changed occurs more than once on the pending line, all occurrences of the string will be replaced by the new string. To selectively change identical strings on the pending line, the set window command can be used. For example, if the pending line were

```
OLD OLD OLD OLD
```

and only the third occurrence of old was desired to be changed, the following commands can be input.

```
/W8,12       <------------------Set window between columns 8 and 12.
/YOLD/NEW    <------------------Exchange OLD with NEW on pending
                                line, display edited line, find next
                                occurrence of OLD, making it pending
                                line and display it.
  OLD OLD NEW OLD          <-----Edited line displayed.
EOF          <------------------End of file.
```

The exchange fields used in the Y command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter (/) is recognized. It is assumed to separate the old string and the new string.

X (Enable Exchange Pattern over Range of Lines, with List)

Used to make string exchanges over a user specified range of lines and then list the lines which have been changed. It is always used in combination with a second command which defines the range of lines over which the string exchange is to take place.

```
Xoldstring/newstring  <------Exchange all occurrences of old string
                             with new string over a specified
                             range of lines within a column window.
                             All changed lines are listed.

range command         <------Range of lines over which exchanges
                             are to take place.
```

```
oldstring
Old string of text which is to be exchanged with new string.  It
can be any length except zero.

newstring
This is the new string of text which is to be exchanged for the
old.  It can be any length including zero.

range command
This is any command which will cause the pending line pointer to
advance through the source file.
```

EXAMPLE:

FILEA contains the following five lines of text:

```
THIS IS AN OLD STRING.
THE OLD STRING CAN BE
REPLACED BY A NEW
STRING WITH THE X
COMMAND.
```

If it is desired to replace all occurrences of the text, STRING, with the text, TEXT, the following commands can be input

```
/XSTRING/TEXT     <------------Exchange all occurrences of STRING
                               with TEXT over the range specified
```

```
                                  in the next command.
/FEOF              <------------Search all lines for STRING and
                                change to TEXT until line with
                                EOF is found in left justified
                                field.
  THIS IS AN OLD TEXT   <------Changed line of text.
  THE OLD TEXT CAN BE   <------Changed line of text.
  TEXT WITH THE X       <------Changed line of text.
EOF                <------------End of file.
```

COMMENTS:

Other examples of range commands would be the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n).

The default print device is the terminal that you input the commands at, but by using the delete lines command (D), the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n), the optional LU command can be invoked to send the changed lines to the printer or a disk file.

The exchange takes place over all occurrences of the old string text within each line unless the window command is used to limit the range of columns over which the exchange is to take place. Using FILEA as an example to demonstrate this the following commands can be input to change only the STRING which starts in column 9 of line two.

```
/W8,15             <------------Set window between columns 8 and 15.
/XSTRING/TEXT      <------------Exchange command.
/+10               <------------Range command. Advance pending
                                line down 10 lines.
  THE OLD TEXT CAN BE   <------Changed line.
EOF                <------------End of file.
```

If additional exchanges are to be made beyond the line positioned to with the range command, the current parameters of the X command may be re-enabled by specifying X with no parameters.

The exchange fields used in the X command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the old string and the new string.

Z (Enable Exchange Pattern over Range of Lines, without List)

Used to exchange a string of text with a new string of text over a specified number of lines. It is always used in combination with a second command which specifies the range of lines over which the exchange is to take place. It is the same as the X command except that there is no listing of the changed lines.

```
Zoldstring/newstring  <------Exchange all occurrences of old string
                             with new string over a specified
                             range of lines within a column window.

range command         <------Range of lines over which exchanges
                             are to take place.
```

oldstring
Old string of text which is to be exchanged with new string. It can be any length except zero.

newstring
This is the new string of text which is to be exchanged for the old. It can be any length including zero.

range command
This is any command which will cause the pending line pointer to advance through the source file.

EXAMPLE:

FILEA contains the following five lines of text:

```
THIS IS AN OLD STRING.
THE OLD STRING CAN BE
REPLACED BY A NEW
STRING WITH THE Z
COMMAND.
```

If it is desired to replace all occurrences of the text, STRING, with the text, TEXT, the following commands can be input

```
/ZSTRING/TEXT       <----------Exchange all occurrences of STRING
```

```
                                  with TEXT over the range specified
                                  in the next command.
/FEOF                <----------Search all lines for STRING and
                                  change to TEXT until line with
                                  EOF is found in left justified
                                  field.
EOF                  <----------End of file.
/1                   <----------Make line one pending line.
/L5                  <----------Display five lines of text.
  THIS IS AN OLD TEXT.
  THE OLD TEXT CAN BE
  REPLACED BY A NEW
  TEXT WITH THE Z
  COMMAND.
EOF                  <----------End of file.
```

COMMENTS:

Other examples of range commands would be the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n).

The exchange takes place over all occurrences of the old string text within each line unless the window command is used to limit the range of columns over which the exchange is to take place. Using FILEA as an example to demonstrate this, the following commands can be input to change only the STRING which starts in column 9 of line two.

```
/W8,15               <----------Set window between columns 8 and 15.
/ZSTRING/TEXT        <----------Exchange command.
/+10                 <----------Range command. Advance pending
                                line down 10 lines.
EOF                  <----------End of file.
/1                   <----------Make line one pending line.
/L5                  <----------Display five lines of text.
  THIS IS AN OLD STRING.
  THE OLD TEXT CAN BE
  REPLACED BY A NEW
  STRING WITH THE Z
  COMMAND.
EOF                  <----------End of file.
```

If additional exchanges are to be made beyond the line positioned to with the range command, the current parameters of the Z command may be re-enabled by specifying Z with no parameters.

The exchange fields used in the Z command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the old string and the new string.

V (Unconditional Character Replace, with List)

Used to insert, delete or replace characters on one or more lines starting from the first column of the edit window (see the SET WINDOW section for details on the W command). It is always used in combination with a second command which defines the range of lines over which the string exchange is to take place. The lines which are changed by this command are displayed to your terminal.

V/newstring	<----------	Insert new string starting at first column of window.
Vxxx/	<----------	Delete number of characters specified by number of characters of xxx starting at first column of window.
Vxxx/newstring	<----------	Replace number of characters specified by number of characters of xxx with new string starting at first column of window.
range command	<----------	Range of lines over which exchanges are to take place.

xxx
Can be any alphanumeric characters since only the number of characters in xxx matters. xxx is not used as a pattern for a search, but rather, to specify the number of characters to be replaced or deleted at the start of the window by the new character string. xxx can be a maximum of 148 alphanumeric characters.

newstring
This is the newstring which is to either replace the number of characters specified by xxx or to be inserted at the first column of the window.

range command
This is any command which will cause the pending line pointer to advance through the source file.

EXAMPLES:

FILEA contains the following three lines of text:

```
AAAAAAAAAAAAAAA
BBBBBBBBBBBBBBB
CCCCCCCCCCCCCCC
```

1) V/newstring example (insert new string).

```
/W5,150            <------------Set window between columns 5 to 150.
/P                 <------------Display pending line.
  AAAAAAAAAAAAAAA    <---------Pending line displayed.
/V/XXX             <------------Insert XXX starting in the first
                                column of the window over the range
                                specified in the next command.
/+2                <------------Advance pending line down two lines.
  AAAAXXXAAAAAAAAAAA <---------Changed line displayed.
  BBBBXXXBBBBBBBBBBB <---------Changed line displayed.
  CCCCCCCCCCCCCCC    <---------New pending line.
```

2) Vxxx/ example (delete characters).

```
/W1,3              <------------Set window between columns 1 to 3.
/P                 <------------Display pending line.
  AAAAAAAAAAAAAAA    <---------Pending line displayed.
/V12345/           <------------Delete first five characters starting
                                in the first column of the window
                                over the range specified in the next
                                command.
/FEOF              <------------Find EOF.
  AAAAAAAAAA       <------------Changed line displayed (first 5
                                characters deleted).
  BBBBBBBBBB       <------------Changed line displayed (first 5
                                characters deleted).
  CCCCCCCCCC       <------------Changed line displayed (first 5
                                characters deleted).
EOF                <------------End of file.
```

3) Vxxx/newstring example (replace characters).

```
/W1,3              <------------Set window between columns 1 to 3.
/P                 <------------Display pending line.
  AAAAAAAAAAAAAAA
/VAA/1111          <------------Replace first two characters starting
                                in the first column of the window
                                over the range specified in the next
                                command with the string 1111.
/+                 <------------Advance pending line one line.
  1111AAAAAAAAAAAAA  <---------Changed line displayed.
  BBBBBBBBBBBBBBB    <---------New pending line.
```

```
/W5,150              <------------Set window between columns 5 to 150.
/VBB/2222            <------------Replace first two characters starting
                                  in the first column of the window
                                  over the range specified in the next
                                  command with the string 2222.
/+                   <------------Advance pending line one line.
  BBBB2222BBBBBBBBB  <----------Changed line displayed.
  CCCCCCCCCCCCCCC    <----------New pending line.
/1                   <------------Make line one pending line and display
                                  it.
  1111AAAAAAAAAAAAA  <----------Pending line displayed.
/L5                  <------------Display five lines of text.
  1111AAAAAAAAAAAAA
  BBBB2222BBBBBBBBB
  CCCCCCCCCCCCCCC
EOF                  <------------End of file.
```

COMMENTS:

The exchange field starts at the first column of the window, but is not limited by the width of the window (note example 2 above).

The default print device is the terminal used to input the commands, but by using the delete lines command (D), the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n), the optional LU parameter in these commands can be invoked to send the changed lines to the printer or a disk file.

The exchange fields used in the V command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the xxx parameter with the new string.

U (Unconditional Character Replacement, without List)

Used to insert, delete or replace characters on one or more lines starting from the first column of the edit window (see the SET WINDOW section for details on the W command). It is always used in combination with a second command which defines the range of lines over which the string exchange is to take place. It is the same as the V command except that there is no listing of the changed lines.

U/newstring	<----------	Insert new string starting at first column of window.
Uxxx/	<----------	Delete number of characters specified by number of characters of xxx starting at first column of window.
Uxxx/newstring	<----------	Replace number of characters specified by number of characters of xxx with new string starting at first column of window.
range command	<----------	Range of lines over which exchanges are to take place.

xxx
Can be any alphanumeric characters since only the number of characters in xxx matters. xxx is not used as a pattern for a search, but rather, to specify the number of characters to be replaced or deleted at the start of the window by the new character string. xxx can be a maximum of 148 alphanumeric characters.

newstring
This is the newstring which is to either replace the number of characters specified by xxx or to be inserted at the first column of the window.

range command
This is any command which will cause the pending line pointer to advance through the source file.

EXAMPLES:

FILEA contains the following three lines of text:

```
AAAAAAAAAAAAAAA
BBBBBBBBBBBBBBB
CCCCCCCCCCCCCCC
```

1) U/newstring example (insert new string).

```
/W5,150            <-------------Set window between columns 5 to 150.
/P                 <-------------Display pending line.
  AAAAAAAAAAAAAAA      <-------Pending line displayed.
/U/XXX             <-------------Insert XXX starting in the first
                                 column of the window over the range
                                 specified in the next command.
/+2                <-------------Advance pending line down two lines.
  CCCCCCCCCCCCCCC      <-------New pending line.
/1                 <-------------Make line one pending line and display
                                 it.
  AAAAXXXAAAAAAAAAAA   <-------Pending line displayed.
/L5                <-------------Display five lines of text.
  AAAAXXXAAAAAAAAAAA
  BBBBXXXBBBBBBBBBBB
  CCCCCCCCCCCCCCC
EOF                <-------------End of file.
```

2) Uxxx/ example (delete characters).

```
/W1,3              <-------------Set window between columns 1 to 3.
/P                 <-------------Display pending line.
  AAAAAAAAAAAAAAA      <-------Pending line displayed.
/U12345/           <-------------Delete first five characters starting
                                 in the first column of the window
                                 over the range specified in the next
                                 command.
/FEOF              <-------------Find EOF.
EOF                <-------------End of file.
/1                 <-------------Make line one pending line and display
                                 it.
  AAAAAAAAAA           <-------Pending line displayed.
/L5                <-------------Display five lines of text.
  AAAAAAAAAA           <-------Changed line displayed (first 5
                                 characters deleted).
  BBBBBBBBBB           <-------Changed line displayed (first 5
                                 characters deleted).
  CCCCCCCCCC           <-------Changed line displayed (first 5
                                 characters deleted).
EOF                <-------------End of file.
```

3) Uxxx/newstring example (replace characters).

```
/W1,3              <-------------Set window between columns 1 to 3.
/P                 <-------------Display pending line.
  AAAAAAAAAAAAAAA      <-------Pending line displayed.
/UAA/1111          <-------------Replace first two characters starting
                                 in the first column of the window
                                 over the range specified in the next
                                 command with the string 1111.
/+                 <-------------Advance pending line one line and
                                 display it.
  BBBBBBBBBBBBBB       <-------New pending line (changed line not
                                 displayed).
/W5,150            <-------------Set window between columns 5 to 150.
/UBB/2222          <-------------Replace first two characters starting
                                 in the first column of the window
                                 over the range specified in the next
                                 command with the string 2222.
/+                 <-------------Advance pending line one line and
                                 display it.
  CCCCCCCCCCCCCC       <-------New pending line (changed line not
                                 displayed).
/1                 <-------------Make line one pending line and display
                                 it.
  1111AAAAAAAAAAAA     <-------Pending line displayed.
/L5                <-------------Display five lines of text.
  1111AAAAAAAAAAAA     <-------Changed line displayed (AA replaced
                                 with 1111 starting in column 1).
  BBBB2222BBBBBBBB     <-------Changed line displayed (BB replaced
                                 with 2222 starting in column 5).
  CCCCCCCCCCCCCC       <-------Unchanged line displayed.
EOF                <-------------End of file.
```

COMMENTS:

The exchange field starts at the first column of the window, but is not limited by the width of the window (note example 2 above).

The exchange fields used in the U command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the xxx parameter with the new string.

A (Abort Edit Session)

Used to abort the edit session and leave the original source file unchanged.

A
No parameters required

EXAMPLE:

FILEA contains the following two lines of text:

```
   THIS EXAMPLE DEMONSTRATES
   THE ABORT COMMAND

   :RU,EDITR            <----------FMGR command to call EDITR.
   SOURCE FILE?         <----------EDITR requesting name of file to
                                   be edited.
   /FILEA               <----------File to be edited.
     THIS EXAMPLE DEMONSTRATES  <-First line of file.
   /-2                  <----------Delete two lines of text.
   EOF                  <----------End of file mark indicates all text
                                   has been deleted.
   /A                   <----------Abort edit session leaving original
                                   file unchanged.
   EDITR ABORTED        <----------EDITR response verifying edit session
                                   has been aborted.
   :LI,FILEA            <----------FMGR command to display contents of
                                   FILEA.

FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001  THIS EXAMPLE DEMONSTRATES
0002  THE ABORT COMMAND
```

EC (End Edit and Create a File Manager File)

Ends the edit session and creates a named File Manager file for storage of the contents of the destination work area.

ECnamr
namr filename[:security code[:cartridge reference]]

EXAMPLE:

```
   :RU,EDITR             <----------FMGR command to run EDITR.
   SOURCE FILE?          <----------EDITR request for name of file to be
                                    edited.
   /0                    <----------Put empty file into EDITR's source
                                    work area.
   EOF                   <----------End of file.
   / LINE ONE            <----------Insert first line of text.
   / LINE TWO            <----------Insert second line of text.
   /EC&FILEA:AA:1000     <----------End edit session and store destination
                                    work area into created file named
                                    &FILEA with security code AA on
                                    cartridge with reference number 1000.
   END OF EDIT           <----------EDITR response indicating edit session
                                    is terminated.
   :LI,&FILEA:AA:1000    <----------FMGR command to list the contents of
                                    file named &FILEA.

   FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002
   0001  LINE ONE
   0002  LINE TWO
```

COMMENTS :

The minimum file size created is two blocks regardless of how few lines are input.

ER (Replace Old File with New File)

Ends an edit session and replaces the FMGR file with the edited version stored in the destination work area.

```
ER              <-------------End edit session and replace disc
                             resident file with edited version.

ERnamr          <-------------End edit session and replace disc
                             resident file with edited version.
```

```
namr
filename[:security code[:cartridge reference]]
```

EXAMPLE:

FILEA contains the following two lines of text:

```
   LINE ONE
   LINE TWO
```

1) ER example.

```
   :RU,EDITR          <-----------FMGR command to run EDITR.
   SOURCE FILE?       <-----------EDITR request for name of file to be
                                  edited.
   /FILEA             <-----------Name of file to be edited.
     LINE ONE         <-----------First line of file.
   /+                 <-----------Advance pending line one line and
                                  display it.
     LINE TWO         <-----------Pending line displayed.
   / LINE THREE       <-----------Insert new line of text.
   /ER                <-----------End edit session and replace contents
                                  of disc resident file with contents of
                                  destination work area.
   END OF EDIT        <-----------EDITR response indicating edit session
                                  has terminated.

   :LI,FILEA          <-----------FMGR command to list the contents of
                                  FILEA to terminal CRT.
```

COMMENTS :

If no room exists on the cartridge for a larger, updated file, an error message is displayed. You should create a new file on another cartridge with sufficient space.

```
   FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

   0001  LINE ONE
   0002  LINE TWO
   0003  LINE THREE
```

2) ERnamr example (Editing a file with a security code).

```
   :RU,EDITR            <----------FMGR command to run EDITR.
   SOURCE FILE?         <----------EDITR request for name of file to be
                                   edited.
   /FILEA               <----------Name of file to be edited.
     LINE ONE           <----------First line of file.
   /+                   <----------Advance pending line one line and
                                   display it.
     LINE TWO           <----------Pending line displayed.
   / LINE THREE         <----------Insert new line of text.
   /ERFILEA:SM          <----------End edit session and replace disc file
                                   named FILEA with edited version
                                   (note that security code, SM, is
                                   required).
   END OF EDIT          <----------EDITR response indicating edit session
                                   has terminated.
```

Chapter 3
Special EDITR Considerations

EDITR in Batch Environment

In batch mode, commands to EDITR are supplied as part of the job command file, that is, all EDITR commands must be supplied before the job is started. Once the job is under way, you cannot interact with the computer to change the processing.

All EDITR commands function the same in batch as they do in the interactive mode. You need not supply the EDITR prompt (/) on the input commands.

It is important that you know beforehand what is going to result from the edit commands in the job deck. Any condition resulting in an error message causes EDITR to abort since user intervention to fix the error is not available in batch. You must, also, ensure that EDITR goes through the normal terminate sequence initiated by EC or ER commands. If an edit reflects this sequence, EDITR aborts.

If EDITR is run in batch mode without spooling, File Manager commands and EDITR commands must be read from an external device. Under spooling, however, EDITR commands can be read from a file if the file looks like an external device. This is done by setting up an input spool to reference a file name as shown in the following examples. The default attributes should always be specified for the input file.

1) EDITR Without Spooling

```
:RU, EDITR, 5<---------Card Reader defined as LU 5.

(Job deck in card reader)
```

2) EDITR With Spooled Job

```
:JO, SMITH
:RU, EDITR, 5
     ^
     |<-------------The EDITR commands appear in the
     |               job stream which may be a file
     |               or a card deck or tape, etc.
     EDITR commands
     |
     v
ECnamr
:EOJ

:JO, SMITH
:LU, 50, COMND<-------Set input spool to reference file
                      COMND containing the EDITR commands.

:RU, EDITR, 50<-------In this example, EDITR acts as if it
                      were reading from LU 50 but it is
                      actually reading records from COMND
                      through the spool monitor driver.

:EOJ
```

For additional information on batch job processing and spooling, refer to the RTE-6/VM Batch and Spooling Reference Manual.

EDITR in a Multi-Terminal Environment

In RTE, both the Sesion Monitor and the Multi-Terminal Monitor, (MTM), allow any terminal user to operate independently of other users. Several users can then run the same program at the same time if each terminal has a separate copy of the program. When the user enters the command to run EDITR

```
:RU, EDITR
```

a program named EDIxx is created and scheduled to run at terminal xx, where xx is the LU of the terminal when operating under the Multi-Terminal Monitor, or the session identification number when operating under the Session Monitor. When EDIxx is finished, the ID segment is automatically returned to the system.

EDITR in a Multipoint Environment

In a multipoint environment, several special considerations apply to EDITR operations. A terminal is defined to be in a multipoint environment if its I/O is being processed through RTE driver DVR07. To determine if a terminal is operating under multipoint, observe the transmit break light on the terminal. If it is blinking intermittently, the terminal is probably operating in a multipoint mode. When EDITR is invoked from a multipoint terminal, several actions are performed for the user:

1. The intrinsic tab function of the terminal is enabled which allows the user to use the TAB key on the terminal to set and clear tab stops with the SET TAB and CLEAR TAB terminal keys. The EDITR's tab character (;) remains on.

2. The INSERT CHAR, DELETE CHAR, and CLEAR DSPLY terminal keys are enabled for editing functions.

3. The Q and O commands, which will be described later in this section, are enabled for character edits.

Functions which do not work in a multipoint environment include:

1. The CNTRL and ESC keys do not work in a multipoint environment, therefore the Q and O commands must be used for character edits.

2. The INSERT LINE and DELETE LINE keys do not work, therefore the EDITR commands to insert and delete lines must be used.

3. The RETURN key (carriage return) is not used for data transmittal in a multipoint environment. The ENTER key is used to transmit data to the multipoint controller.

Furthermore, the characters that EDITR will process are usually delimited by the left margin on the left and the current cursor position on the right.

Except for the differences cited above, all of the other EDITR commands may be used in a multipoint environment.

Character Edits With the Q and O Commands

The Q and O Commands are used for character edits in a multipoint environment. Both commands are used to edit the pending line. The only difference between them is that the Q command edits the pending line, whereas the O command duplicates the pending line and then edits the duplicate. Although only the Q command is discussed in this section, the discussion applies equally to the O command.

When the Q or O command is entered, the pending line is displayed along with a delimiter (GS) to the left of the line. The delimiter is not part of the text string but must be preserved to assure proper operation. The delimiter is represented as a (#) pound sign throughout the rest of this section. EDITR will position the cursor under the first character of the line. To retain the pending line as is, immediately press the ENTER key as follows:

```
/Q
 #ABCDE<------------Cursor displayed under initial
  _                 character,press the ENTER key
                     and line remains unchanged.
/P<-----------------Display pending line.
  ABCDE<------------Pending line displayed.
```

Delete Characters from the End of a Line

To truncate characters from the end of a line, position the cursor immediately after the last character to be retained. Press the ENTER key to enter all the characters between the left margin and the selected cursor position. The CLEAR DSPLY terminal key may then be used to delete all remaining characters to the right of the cursor. Following the use of the CLEAR DSPLY key, the ENTER key is used to enter the then edited line as follows:

```
/Q<-----------------Enter Q to make character edit.
 #ABCDE<------------Pending line is displayed, cursor
     _              is moved under the D using the
                     right arrow key (->). Clear DSPLY
                     key may then be used to clear D
                     and E from the screen.
  ABC<--------------Edited line is displayed,
                     ENTER key may be used to
                     enter the edited line.
```

Insert Characters Within a Line

The INSERT CHAR key may be used to add characters anywhere in a line. Position the cursor to the appropriate column and line, press the INSERT CHAR key noting the red status light immediately above the key. Type in the new characters and upon completion, move cursor to the end of the line where the ENTER key is pressed. The INSERT CHAR status light will go off and the edited line will remain as the new pending line, for example:

```
/Q <-----------------Enter Q to make character edit.
 #ABCDE<-------------Pending line is displayed, the
    _                cursor is positioned beneath C
                      using the right arrow key (->).
                      INSERT CHAR key is pressed
                      and XXX is entered.
  ABCXXXDE<-----------------Edited line is displayed.
```

Delete Characters Within a Line

To delete one or more characters, position the cursor under the initial character to be deleted and press the DELETE CHAR key. The character will be deleted from the display and and the remainder of the line will shift left to fill the gap. After all desired deletions have been made, move the cursor to the end of the line and depress the ENTER key. Do not delete the delimiter at the beginning of the line. For example:

```
/Q<------------------Enter Q to make character edit.
 #ABCDE<-------------Pending line displayed, cursor
   _                 is moved beneath B using the
                      right arrow key (->). DELETE
                      CHAR key is pressed three
                      times to sucessively delete B, C,
                      and D. When the required number
                      of characters have been deleted,
                      position the cursor at the
                      end of the line and press
                      the ENTER key.
 AE<-----------------Edited line is displayed.
```

Tab Control in Multipoint Environment

When EDITR is invoked in a multipoint environment, the TAB key on the terminal is enabled (the EDITR tab character remains on). The tab stops are initially set to columns 7 and 21 but may be changed using SET TAB and CLEAR TAB keys (in the terminal Display Function). The TAB key may be used to position the cursor at any time. It is equivalent to moving the cursor using any of the five directional arrow keys in the terminal Display Group.

Appendix A
EDITR Messages

Errors encountered by the Editor and information messages are reported on the operator console. A summary of the messages, their effects, and error recovery procedures are contained in Table A-1:

Table A-1. EDITR Message Summary

MESSAGE	DESCRIPTION	SUGGESTED ACTION
EDITR ABORTED	If this message is displayed after you press "A", to abort EDITR, there is no error. Otherwise, the RTE or Batch-Spool Monitor systems have aborted EDITR. The source file remains unchanged, the destination file is deleted.	Rerun EDITR
??	Syntax error in the command just given to EDITR or the input device has timed out waiting for a command.	Retype the command
EOF	A command has caused an attempt to read beyond the current end of the source file.	Return to start of file by typing in a zero or one.

MANUAL PART NO. 92084-90017
Printed in U.S.A. December 1981

HEWLETT-PACKARD COMPANY
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014