IBM

Virtual Machine/System Product

SC24-5221-05

System Product Editor Command and Macro Reference

Release 6

IBM

SC24-5221-05

Virtual Machine/System Product

System Product Editor Command and Macro Reference

Release 6

Acknowledgement

We gratefully acknowledge the permission to reprint excerpts from the following:

The People's Almanac, by David Wallechinsky and Irving Wallace. Copyright © 1975 by David Wallace and Irving Wallace. Reprinted by permission of Doubleday & Company, Inc.

I Wouldn't Have Missed It, by Ogden Nash, reprinted by permission of Curtis Brown, Ltd.

Sixth Edition (August 1988)

This edition, SC24-5221-05, is a major revision of SC24-5221-04 and applies to the Virtual Machine/System Product Release 6, program number 5664-167, and to all subsequent releases of this product until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a list of changes, see "Summary of Changes" on page 455.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Preface

Use this book as a reference manual; it contains all of the command formats, syntax rules, and operand and option descriptions for the XEDIT command and XEDIT subcommands and macros. For tutorial information on using the editor, refer to the *VM/SP System Product Editor User's Guide*. The *User's Guide* also contains information on using the System Product Interpreter, which processes programs written in the Restructured Extended Executor (REXX) language, for XEDIT macros. You should be familiar with the *User's Guide* before you attempt to use this reference book. This publication has the following chapters:

"Chapter 1: Rules and Conventions" tells you how to enter the XEDIT command and its subcommands and macros. It lists the notation conventions used in this book, so that you can interpret the command format descriptions starting in Chapter 2.

"Chapter 2: The XEDIT Command" contains the format description, and operand and option list for the XEDIT command, which is used to invoke the editor.

"Chapter 3: XEDIT Subcommands and Macros" describes the subcommands and macros available in the environment of the editor. Each subcommand and macro description contains usage notes and/or examples, summarizes the types of responses you might receive, and lists the error messages and return codes.

"Chapter 4: XEDIT Prefix Subcommands and Macros" describes the prefix subcommands and macros, which are entered directly in an area called the prefix area of any line in a file.

This book also has the following appendixes:

"Appendix A: File Type Defaults" lists the special file types that are recognized by the editor and indicates the default settings that the editor supplies for logical record length, logical tabs, truncation, and so on.

"Appendix B: Effects of Selective Line Editing Subcommands" details the effects of SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW on other XEDIT subcommands.

"Appendix C: CMS Editor (EDIT) Migration Mode" explains how to edit a file in EDIT migration mode.

"Appendix D: Migrating from EDIT to XEDIT" lists the EDIT subcommands and their XEDIT counterparts.

"Appendix E: Optimizing Macros" describes how to improve the performance of macros and lists those XEDIT macros that have been optimized.

"Appendix F: Using Double-Byte Character Sets" describes the special considerations for manipulating DBCS strings in the XEDIT environment.

"Appendix G: XEDIT Virtual Screens and Windows" describes how virtual screens and windows are defined.

"Appendix H: A Summary of XEDIT Subcommands and Macros" lists all the XEDIT subcommands and macros, their abbreviations, and a brief description of functions.

Contents

Chapter 1. Rules and Conventions

XEDIT subcommands and macros follow the same rules and conventions. For purposes of this discussion, "subcommand" refers to both XEDIT subcommands and XEDIT macros.

The general format of XEDIT subcommands is:

subcommand name	*operands...*

At least one blank must separate the subcommand name and the operands, unless the operand is a number or a special character. For example, NEXT8 and NEXT 8 are equivalent.

At least one blank must be used to separate each operand in the command line unless otherwise indicated.

The maximum length of an XEDIT subcommand issued from an exec procedure or from an XEDIT macro is 256 characters. The maximum length of an XEDIT subcommand issued from the XEDIT command line is 255 characters.

Subcommand Name

The subcommand name is an alphabetic symbol of one-to-eight characters. In general, the names are based on verbs that describe the function you want the editor to perform. For example, the ADD subcommand adds lines to the file.

Subcommand Operands

The subcommand operands may be either keyword or positional or a combination of both. The operands specify the information on which the editor operates when it performs the subcommand function. The operands must be entered in the order in which they appear in the command format boxes.

One of the most widely-used operands in XEDIT is the "target" operand, which provides various ways to identify a line to the editor. The concept of a target is described in the LOCATE subcommand in this book and in the *VM/SP System Product Editor User's Guide*. You should become familiar with targets before attempting to use XEDIT subcommands that require target operands.

Character Set Usage

XEDIT subcommands may be entered using a combination of characters from six different character sets. The contents of each of the character sets is shown in Table 1 on page 2.

Table 1. Character Sets and Their Contents

Character Set	Names	Symbols
Separator	Blank	
National	Dollar Sign Pound Sign At Sign	$ # @
Alphabetic	Uppercase Lowercase	A - Z a - z
Numeric	Numeric	0 - 9
Alphameric	National Alphabetic	$, #, @ A - Z a - z
	Numeric	0 - 9
Special		All other characters

Notation Conventions

The notation used to define the command syntax in this book is:

- Abbreviations

  Where an abbreviation of a subcommand name is permitted, the shortest acceptable version of the name is represented by uppercase letters. (However, the subcommands can be entered on the terminal in any combination of uppercase and lowercase letters.) For example, the subcommand

  ```
  DELete
  ```

  may be specified as DEL, DELE, DELET, or DELETE.

  Operands are represented in the same way. When an abbreviation is permitted, the shortest acceptable version of the operand is represented by uppercase letters in the subcommand format box. If no minimum abbreviation is shown, the entire word (represented by uppercase letters) must be entered.

- The following symbols are used to define the subcommand format and should never be typed when the actual subcommand is entered.

underscore	_
braces	{ }
brackets	[ ]
ellipsis	...

- Uppercase letters and the following symbols should be entered as specified in the format box.

asterisk	*
comma	,
equal sign	=
parentheses	( )
period	.
colon	:

- Lowercase letters, words, and symbols that appear in the subcommand format box represent variables for which specific information must be substituted when the subcommand is entered. For example, "*fn ft fm*" indicates that a file identifier such as "MYFILE SCRIPT A1" should be entered.
- Brackets around a single operand mean the operand is optional.

  ```
  FILE [fn]
  ```

  The "*fn*" operand is optional.

  ```
  FILE fn
  ```

  The "*fn*" operand *must* be entered.
- Choices are represented in the format boxes by stacking the operands or by separating the operands with a vertical bar (|). For example,

  $$\begin{matrix}A\\B\\C\end{matrix}\ \underline{\text{or}}\ A|B|C$$

  indicates that a choice is to be made between A, B, and C.

  Braces also indicate that a choice is to be made. For example,

  $$\left\{\begin{matrix}A\\B\\C\end{matrix}\right\}\ \underline{\text{or}}\ \{A|B|C\}$$

  indicates that a choice is to be made between A, B, and C.
- The use of brackets denotes choices, one of which *may* be selected. For example,

  $$\begin{bmatrix}A\\B\\C\end{bmatrix}\ \underline{\text{or}}\ [A|B|C]$$

  indicates that you may enter A, B, or C, or you may omit the operand.
- An underscored operand represents a default. If the operand is omitted, the editor automatically supplies the operand that is underlined. For example,

  $$\begin{bmatrix}A\\\underline{B}\\C\end{bmatrix}$$

  If no operand is specified, B is assumed.

- An ellipsis indicates that the preceding operand may be repeated successively. For example,

```
A
B ...
C
```

  indicates that you must select A, B, or C one time and that you may specify one of the three more than once in succession, for example,

```
A B C A
```

Chapter 2. The XEDIT Command

Use the CMS command XEDIT to invoke the editor to create, modify, and manipulate CMS files on disk or in Shared File System (SFS) directories. Once the editor has been invoked, you may execute XEDIT subcommands and use the System Product Interpreter or EXEC 2 macro facility.

You can return control to the CMS environment by entering the XEDIT subcommand FILE or QUIT.

Format

```
Xedit    [fn [ft [fm]]]     [(options...[)] ]

         Options:
           [Width nn]  [NOSCreen]  [PROFile macroname]
           [NOPROFil]  [NOCLear]   [NOMsg]
           [MEMber membername]   [WINdow wname]

           [LOCk   ]
           [NOLOCk ]

         Options Valid Only in Update Mode:

           [Update  ]   [Seq8  ]   [Ctl fn1]
           [NOUpdate]   [NOSeq8]   [NOCtl  ]

           [Merge]   [UNtil filetype]   [Incr nn]
           [SIDcode string]
```

Operands

fn ft
: are the file name and the file type of the file to be edited. If they are not specified here, they must be provided in the LOAD subcommand as part of the profile.

fm
: is the file mode of the file to be edited, indicating an accessed minidisk or SFS directory where the file resides. The editor determines the file mode of the edited file as follows:

- Editing existing files.

  When the file mode is specified, that disk or directory and its extensions are searched. If the file mode is not specified or is specified as an asterisk (*), all accessed disks and/or directories are searched for the specified file.

- Creating new files.

  If the file mode is not specified, the editor assumes a file mode of A1.

Options

Width *nn*
defines the amount of virtual storage used to contain one line of the file. If the value specified is too small, certain file lines may be truncated.

If not specified here, WIDTH may be defined in the LOAD subcommand as a part of the profile. If WIDTH is not specified in either the XEDIT command or the LOAD subcommand, the default is the larger of the following:

- The logical record length (LRECL) of the file
- The default logical record length associated with the file type. See Appendix A for a list of these defaults.

NOSCreen
forces a 3270 display terminal into line (typewriter) mode.

PROFile *macroname*
If the specified macro exists on one of the accessed disks or SFS directories, the editor executes it as the first subcommand.

If the specified macro is not found on an accessed CMS disk or SFS directory, an error message is displayed.

If this option is not specified but a macro with a macro name of PROFILE exists, the editor executes it.

In all cases, the profile macro must have a file type of XEDIT.

NOPROFil
forces the editor not to execute the default PROFILE macro.

NOCLear
specifies that the screen is not cleared when the editor gets control. Instead, the screen is placed in a MORE... (waiting) status. Any messages remain on the screen until the CLEAR key is pressed. This option is useful when the XEDIT command is issued from a macro that displays messages.

NOMsg
enters a file with a default of SET MSGMODE OFF.

MEMber *membername*
is the name of a member in the macro library specified in fn ft fm. If MEMBER is specified, ft must be MACLIB. When the MEMBER option is specified, XEDIT scans the specified MACLIB to find the member. If the member is found, XEDIT reads it into storage. If the member does not exist in that library, a new file is created with a file ID of 'membername MEMBER fm'.

WINdow *wname*
is the name of the virtual screen and window that XEDIT will use to display the file being edited.

Note: The window name must not contain invalid file name characters. Also, "CMS" can not be used as the window name.

For more information on virtual screens and windows, see Appendix G in this book.

LOCk
causes the editor to lock the file to prevent other users from modifying the file while you are editing it. Only existing files in SFS directories can be locked; the LOCK option is ignored for files on minidisks. You must have write authority to the file to lock it; if you have only read authority a warning is displayed and the editing session continues without locking the file. LOCK is the default.

The type of lock XEDIT uses is an update session lock. The file will be locked only for the duration of your editing session. Other users will be able to read the file while it is locked, but only you will be able to write to it.

NOLOCk
indicates that you do not want the editor to try to lock the file. This option can be used to edit a file that another user has locked SHARE or UPDATE. If you specify NOLOCK, other users may change the file while you are editing it. The NOLOCK option is ignored for files on minidisks.

You should only use this option if you are not going to make any changes to the file, or if you will save your changes under a different file identifier. Otherwise, any changes that you make will not include modifications made to the permanent copy of the file by other users during your editing session.

The following options are meaningful only if XEDIT is to be used in update mode:

Update
The editor searches all accessed CMS minidisks and SFS directories for a file with a file name of fn and a file type of UPDATE. If the file exists, the editor applies the update statements before displaying the file to be edited. Each new modification made by the user is added to the existing UPDATE file. The original source file is *not* modified.

If the file does not exist, the editor creates a new UPDATE file with a file mode of A1 to contain modifications made by the user.

NOUpdate
specifies that the editor is to apply no update statements (even if UPDATE is specified in the LOAD subcommand in the profile).

Seq8
specifies that the entire sequence field (the last eight columns of each file line) contains an eight-digit sequence number. The SEQ8 option automatically forces the UPDATE option. SEQ8 is the default value.

NOSeq8
specifies that the last eight columns of the file line contain a three-character label field, followed by a five-digit sequence number.

The NOSEQ8 option forces the UPDATE option.

Ctl *fn*1
specifies that "fn1 CNTRL" is an update control file that controls the application of multiple update files to the file to be edited. (See the CMS UPDATE command in the *VM/SP CMS Command Reference* for more information.)

This option automatically forces the UPDATE and SEQ8 options.

NOCtl
specifies that the editor is not to use the control file (even if it is specified in the LOAD subcommand in the profile).

Merge
specifies that all the updates made through the control file and all the changes made while editing will be written into the file whose name is defined by the latest update level (that is, the most recently applied UPDATE file in a control file). This option forces the UPDATE option.

UNtil *filetype*

specifies the file type of the last update to be applied to the file. Changes are applied to the file being edited from all file types in the control file, up to and including the file type specified in the UNTIL option.

File types of update files listed in the control file or of update files listed in an auxiliary control file can be specified with the UNTIL option. AUX file types (AUXxxxxx) cannot be specified with the UNTIL option.

The UNTIL option forces the UPDATE option.

Incr *nn*

When inserting new lines in an update file, the editor automatically computes the serialization; the INCR option forces a minimum increment between two adjacent lines. If this option is not specified, the minimum increment is one (1). This option forces the UPDATE option.

SIDcode *string*

specifies a string that the editor inserts in every line of an update file, whether the update file is being created or is an existing file. The editor inserts the specified string in the first eight columns of the last 17 columns of the file line (lrecl-16 to lrecl-9). For example, if you have a file with fixed, eighty-character lines, the editor inserts the string in columns 64-71. If the string is less than eight characters, it is padded on the right with blanks. Any data in the eight columns is overlaid.

This option forces the UPDATE option.

Usage Notes

1. In order to use XEDIT on files in SFS directories, the directories must be accessed and you must have read or write authority to the file.
2. When the LOCK option is in effect, it is possible for the lock to be removed during your edit session if one of the following abnormal errors occurs:
   - file system server failure
   - network or APPC/VM failure on the last communication link with the file pool.

   Note: If a CMS abend occurs, the update session lock obtained by XEDIT will **not** be deleted.
3. For the PROFILE, CTL, SIDCODE, INCR, UNTIL, MEMBER, and WIDTH options, the operand must be specified; otherwise, the next option will be interpreted as its operand. For example, in the "PROFILE macroname" option, "macroname" must be specified; if it is not, the next option will be interpreted as the operand "macroname."
4. Once the XEDIT *command* has been executed, the XEDIT *subcommand* can be used to edit and display multiple files simultaneously (see the XEDIT subcommand).
5. You can also call the editor recursively (using "CMS XEDIT. . .," for example). This ability is particularly useful when applications are developed using the editor and its macro facilities to interface with the user, for example, HELP.
6. The editor is kept in virtual storage as part of the CMS nucleus shared segment; the CMS user area is unused. As a result, assuming a large enough virtual machine, any CMS or CP command may be issued directly from the editor environment itself (if a SET IMPCMSCP subcommand is in effect).

When you call CMS XEDIT recursively, a new ring of files is begun that is independent of any previous ring(s).

7. When the PROFILE macro is invoked by an XEDIT command, everything following the command name XEDIT is tokenized (truncated to eight characters), and then assigned to the argument string that is passed to the PROFILE macro.

   The editor does not examine any parameters that follow a closing right parenthesis on the XEDIT command.

8. When you issue an XEDIT command for a variable-format file, trailing blanks are removed when the file is filed (or saved).

9. Comment control records are deleted from an update file whenever an update file is applied to the original source file during an editing session, and a FILE or SAVE subcommand is issued.

10. Many languages have more characters than can be displayed using one-byte codes (KANJI, for example). A Double-Byte Character Set (DBCS) is used to represent those characters. The double-byte characters can appear in a sentence with characters from other languages that are displayed in 1-byte codes. Files containing double-byte characters are handled differently than files that only contain 1-byte characters.

11. The MEMBER option and the NOUPDATE option have no effect when preceded by an option that automatically forces update processing. Likewise, options that usually force update processing are ignored when preceded by the MEMBER option or the NOUPDATE option.

12. If full-screen CMS is set ON before XEDIT writes to the screen, XEDIT will issue the command SHOW WINDOW CMSOUT followed by SHOW WINDOW XEDIT or a SHOW for the particular window that has been set up to display the file.

Responses

When editing a file that resides in an SFS directory, if you have only read authority to the file and you do not specify the NOLOCK option, you will receive the message:

```
1299W Warning: not authorized to lock fn ft fm
```

The editing session will continue but the file will not be locked.

The following messages are displayed only if you are using XEDIT in update mode:

```
178I Updating fn ft   fm
     Applying fn ft   fm
          .
          .
          .
180W Missing PTF file fn ft fm
```

Messages

002E	File *fn ft fm* not found [RC=28]
003E	Invalid option: *option* [RC=24]
024E	File XEDTEMP CMSUT1 A1 already exists [RC=28]
029E	Invalid parameter *parameter* in the option *option* field [RC=24]
048E	Invalid mode *mode* [RC=24]
054E	Incomplete fileid specified [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
065E	*option* option specified twice [RC=24]
066E	*option1* and *option2* are conflicting options [RC=24]
069E	Filemode *mode* not accessed [RC=36]
070E	Invalid parameter *parameter* [RC=24]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, or 100]
109S	Virtual storage capacity exceeded [RC=104]
132S	File *fn ft fm* too large [RC=88]
229E	Unsupported OS dataset, error *nn* [RC=80, 81, 82, or 83]
500E	Unable to unpack file *fn ft fm* [RC=88]
508E	LOAD must be the first subcommand in the profile [RC=3]
554E	Not enough virtual storage available [RC=104]
571I	Creating new file:
622E	Insufficient free storage for {MSGLINE|PFkey/PAkey|synonyms}
915E	Maximum number of windows already defined [RC=13]
927E	The virtual screen must contain at least 5 lines and 20 columns [RC=24]
928E	Command is not valid for virtual screen *CMS* [RC=12]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1214W	File *fn ft fm* already locked SHARE
1215E	File *fn ft fm* is locked {SHARE|UPDATE|EXCLUSIVE} by another user [RC=70]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 70, 76, 99, or 100]
1299W	Warning: Not authorized to lock file *fn ft fm*
1300E	Error *nn* {locking|unlocking} file *fn ft* {*fm*|*dirname*} [RC=55, 70, 76, 99, or 100]

Messages with Member Option

007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
033E	File *fn ft fm* is not a library [RC=32]
039E	No entries in library *fn ft fm* [RC=32]
167S	Previous MACLIB function not finished [RC=88]
622E	Insufficient free storage for reading map [RC=104]

Messages with Update Options

007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
007E	File *fn ft fm* does not have a logical record length greater than or equal to 80 [RC=32]
007E	File *fn ft fm* does not have the same format and record length as *fn ft fm* [RC=32]
007E	File *fn ft fm* is not fixed record format [RC=32]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 32, or 55]
174W	Sequence error introduced in output file: *seqno1* to *seqno2* [RC=32]
178I	Applying *fn ft fm*
179E	Missing or invalid MACS card in control file *fn ft fm*
180W	Missing PTF file *fn ft fm*

183E	Invalid {CONTROL|AUX} file control card [RC=32]
184W	./ S not first card in update file--ignored [RC=32]
185W	Non numeric character in sequence field *seqno* [RC=32]
186W	Sequence number [*seqno1*] not found [RC=32]
207W	Invalid update file control card [RC=32]
210W	Input file sequence error: *seqno1* to *seqno2* [RC=32]
570W	Update *ft* specified in the UNTIL option field not found
597E	Unable to merge updates containing ./S cards [RC=32]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 70, 76, 99, or 100]

Return Codes

0	Normal
3	LOAD must be the first subcommand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Command is not valid for virtual screen
13	Maximum number of windows already defined
20	Invalid character in file name or file type
24	Invalid parameters or options
28	Source file not found (UPDATE MODE), or library not found (MEMBER option), or specified PROFILE macro does not exist, or file XEDTEMP CMSUT1 already exists
31	A rollback occurred
32	Error during updating process, or file is not a library, or library has no entries, or file is not fixed, 80 char. records
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
88	File is too large and does not fit into storage, or previous Maclib function was not finished
99	A required system resource is not available
100	Error reading the file into storage
104	Insufficient storage available

Chapter 3. XEDIT Subcommands and Macros

This chapter describes the formats and operands of the XEDIT subcommands and macros. XEDIT subcommands and macros are valid only in the environment of the editor, which is invoked with the CMS command XEDIT, described in "Chapter 2: The XEDIT Command."

The editor has two modes of operation: edit mode and input mode. Whenever the XEDIT command is entered, edit mode is entered; when the INPUT or REPLACE subcommands are issued with no operands, or when the POWERINP subcommand is issued, input mode is entered.

For tutorial information on how to use the editor, refer to the *VM/SP System Product Editor User's Guide.*

The XEDIT subcommands and macros are listed in alphabetical order for easy reference. Each subcommand and macro description includes the format and description of operands and, where applicable, usage notes, notes for macro writers, responses, error messages and return codes, and examples.

The following commands and subcommands exist in the CP, CMS, and XEDIT environments:

CP
QUERY
SET.

The following commands and subcommands exist in the CMS and XEDIT environments:

HELP
LOAD
SORT
XEDIT.

The following command and subcommand exists in the CP and XEDIT environments:

RESET.

ADD

Use the ADD subcommand to insert blank lines immediately after the current line.

Format

Add	[*n* |<u>1</u>]

Operand

n

is the number of blank lines you want to add. If you omit n, one line is added.

Usage Notes

1. You can enter data in the newly-added lines at any time during the editing session. These blank lines remain in the file after it is saved or filed.
2. If the current line is the End of File line, the lines are added preceding this line.

Responses

If SET IMAGE ON is in effect, the cursor moves to the first tab column of the first line that was added. Otherwise, it is placed in column 1.

By default, the prefix areas associated with the added lines are highlighted. For more information, refer to SET COLOR PENDING.

Each line that is added is prefilled with the current mask (see SET MASK).

The line pointer remains unchanged.

Messages

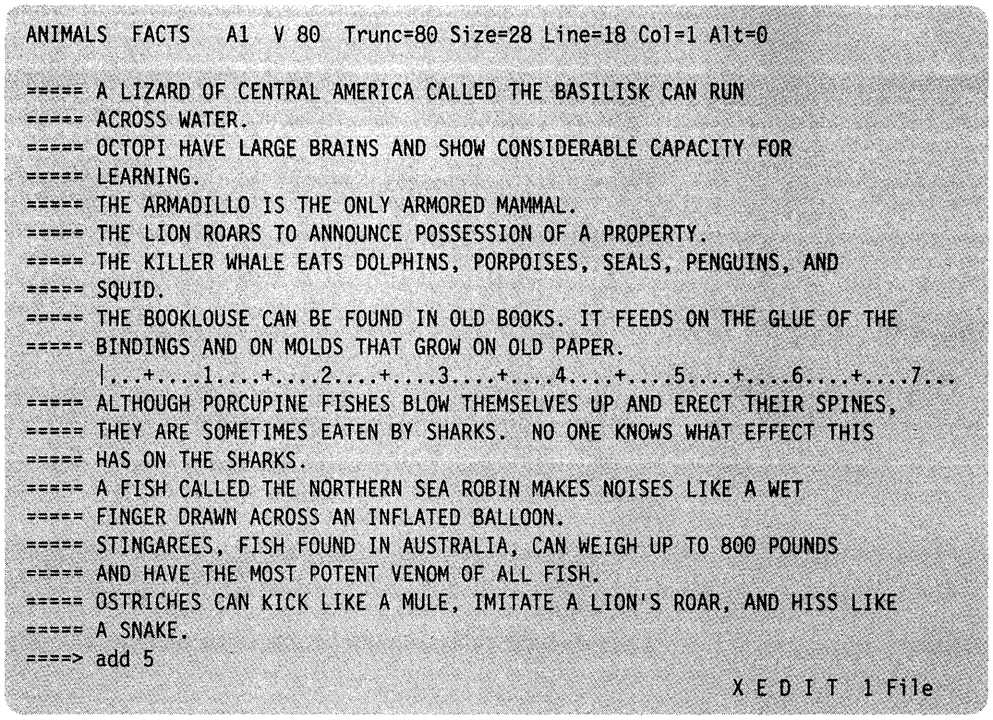
520E	Invalid operand: *operand* [RC=5]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
543E	Invalid number: *number* [RC=5]
557S	No more storage to insert lines [RC=4]

Return Codes

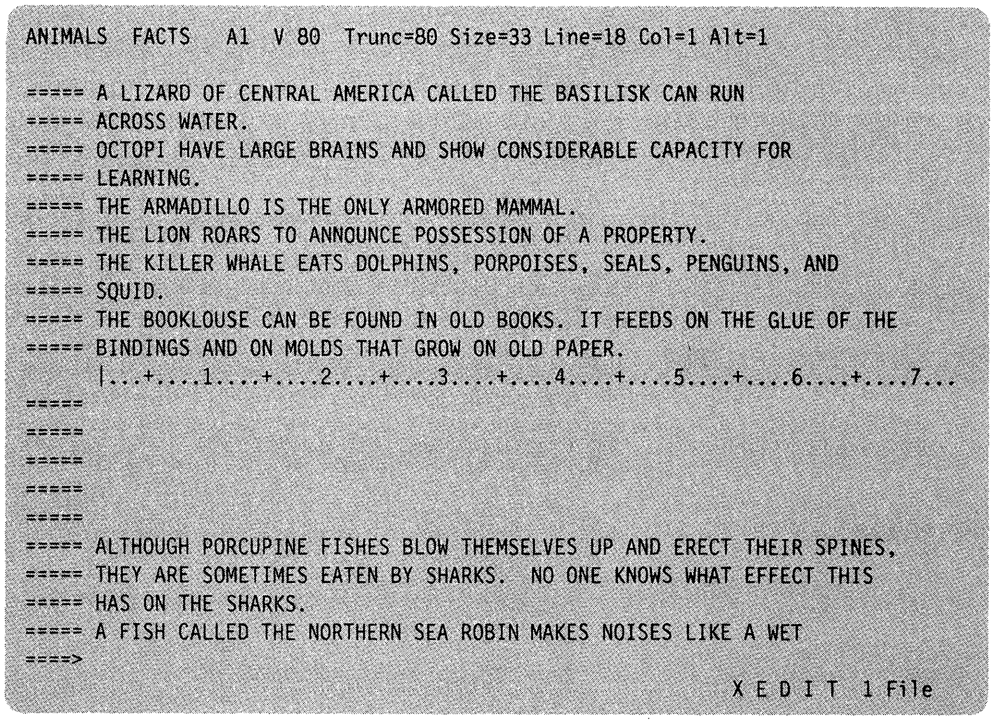
0	Normal
3	Terminal is not a display
4	Insufficient storage to add lines
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 1 is a before-and-after example of the ADD subcommand.



```
ANIMALS  FACTS   A1  V 80  Trunc=80 Size=28 Line=18 Col=1 Alt=0

===== A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
===== THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS. IT FEEDS ON THE GLUE OF THE
===== BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
===== HAS ON THE SHARKS.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
===== FINGER DRAWN ACROSS AN INFLATED BALLOON.
===== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
===== AND HAVE THE MOST POTENT VENOM OF ALL FISH.
===== OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISS LIKE
===== A SNAKE.
====> add 5
                                                              X E D I T  1 File
```



```
ANIMALS  FACTS   A1  V 80  Trunc=80 Size=33 Line=18 Col=1 Alt=1

===== A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
===== THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS. IT FEEDS ON THE GLUE OF THE
===== BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====
=====
=====
=====
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
===== HAS ON THE SHARKS.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
====>
                                                              X E D I T  1 File
```

Figure 1. The ADD Subcommand — Before and After

ALL (Macro)

Use the ALL macro to display a specified collection of lines for editing, while excluding others from display. The collection is specified by a target that is repeatedly applied to the entire file, starting at the top of the file (or range).

Format

ALL	[*rtarget*]

Operand

rtarget
is a target that defines which lines are displayed. The target is "repeated," that is, it is applied from the top of the file (or range) for as many times as necessary to collect all the lines in the file that correspond to the specified rtarget. For example, ALL/KEN/ displays all lines in the file that contain the string "KEN". If no rtarget is specified, ALL displays the entire file.

An rtarget may be specified as an absolute line number, a relative displacement, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*

Usage Notes

1. After you've used the ALL macro to make changes to selected lines in a file, you can redisplay the entire file, including the changes, by issuing the ALL macro by itself with no rtarget specified. All lines in the file are set to a selection level of 0 and DISPLAY is set to 0 0.
2. ALL modifies the SELECT setting of all of the lines in the file and overrides the DISPLAY and SCOPE settings. ALL sets the selection level of all selected lines to 1 and all nonselected lines are set to 0. After ALL is specified, the DISPLAY setting is set to 1 1 and SCOPE is set equal to DISPLAY. If you have used the SCREEN option, the DISPLAY setting is 1 1 for the view for which the ALL subcommand was entered, and unchanged for all other views. (Refer to SET SELECT, SET DISPLAY, SET SCOPE, and SET SCREEN in this publication.)
3. ALL does not change the SHADOW setting. (Refer to SET SHADOW in this publication.) The following example illustrates how ALL performs when SET SHADOW is "ON" (the default). When SET SHADOW is "OFF," then no notice is displayed to indicate that lines are not displayed.

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]

Return Codes

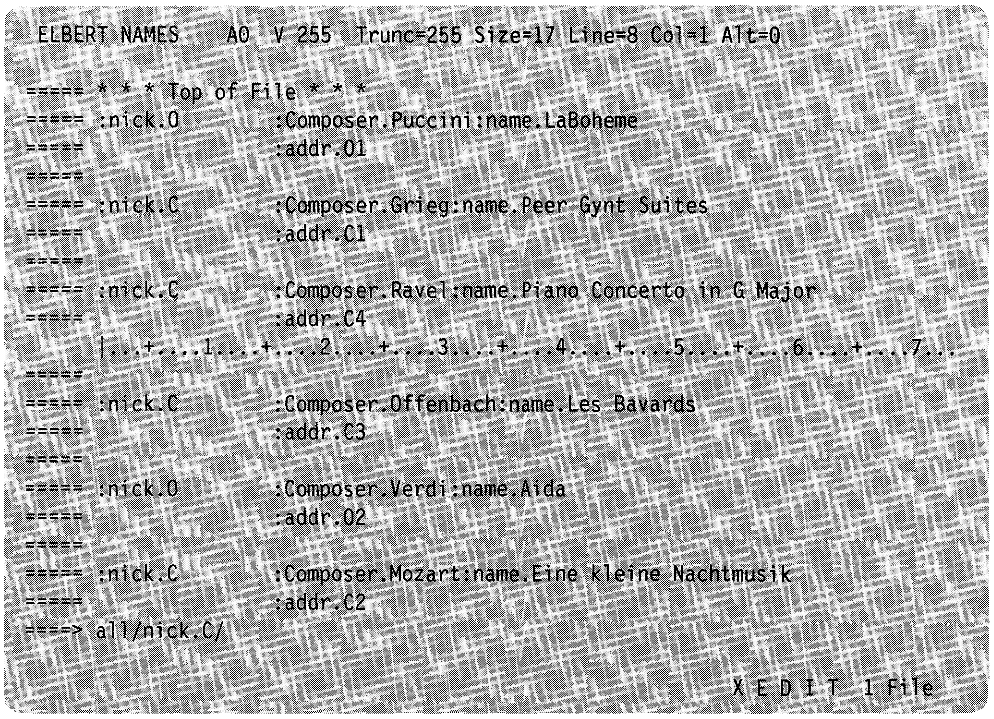
0	Normal
2	Target not found
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 2 is an example of the ALL Macro.

Elbert had a record collection for which he established a "names" file. He used the following organization scheme:

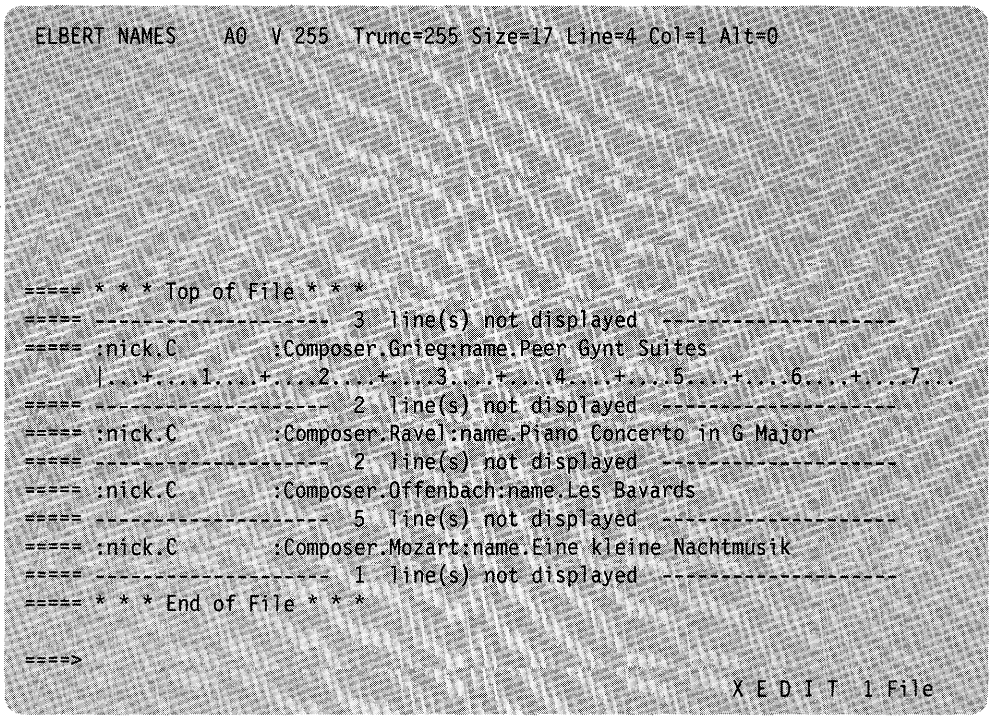
```
NICK      O for opera
          C for classical

COMPOSER  Name of Composer

NAME      Name of Composition

ADDRESS   C or O, depending on type, followed by the
          assigned number on the jacket.
```



```
 ELBERT NAMES    A0  V 255  Trunc=255 Size=17 Line=8 Col=1 Alt=0

===== * * * Top of File * * *
===== :nick.O         :Composer.Puccini:name.LaBoheme
=====                 :addr.O1
=====
===== :nick.C         :Composer.Grieg:name.Peer Gynt Suites
=====                 :addr.C1
=====
===== :nick.C         :Composer.Ravel:name.Piano Concerto in G Major
=====                 :addr.C4
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
===== :nick.C         :Composer.Offenbach:name.Les Bavards
=====                 :addr.C3
=====
===== :nick.O         :Composer.Verdi:name.Aida
=====                 :addr.O2
=====
===== :nick.C         :Composer.Mozart:name.Eine kleine Nachtmusik
=====                 :addr.C2
====> all/nick.C/

                                                            X E D I T  1 File
```

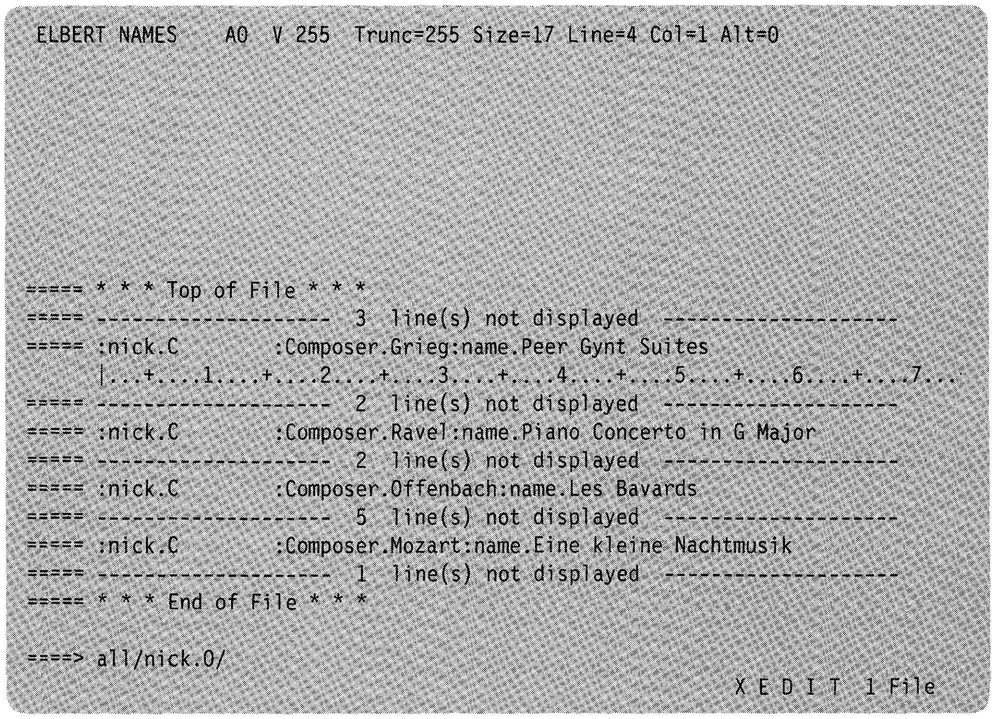
Using the ALL macro to select and display all classical records in Elbert's collection

Figure 2 (Part 1 of 3). The ALL Macro



```
 ELBERT NAMES    A0  V 255  Trunc=255 Size=17 Line=4 Col=1 Alt=0








===== * * * Top of File * * *
===== -------------------- 3  line(s) not displayed --------------------
===== :nick.C       :Composer.Grieg:name.Peer Gynt Suites
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== -------------------- 2  line(s) not displayed --------------------
===== :nick.C       :Composer.Ravel:name.Piano Concerto in G Major
===== -------------------- 2  line(s) not displayed --------------------
===== :nick.C       :Composer.Offenbach:name.Les Bavards
===== -------------------- 5  line(s) not displayed --------------------
===== :nick.C       :Composer.Mozart:name.Eine kleine Nachtmusik
===== -------------------- 1  line(s) not displayed --------------------
===== * * * End of File * * *

====>
                                                      X E D I T  1 File
```

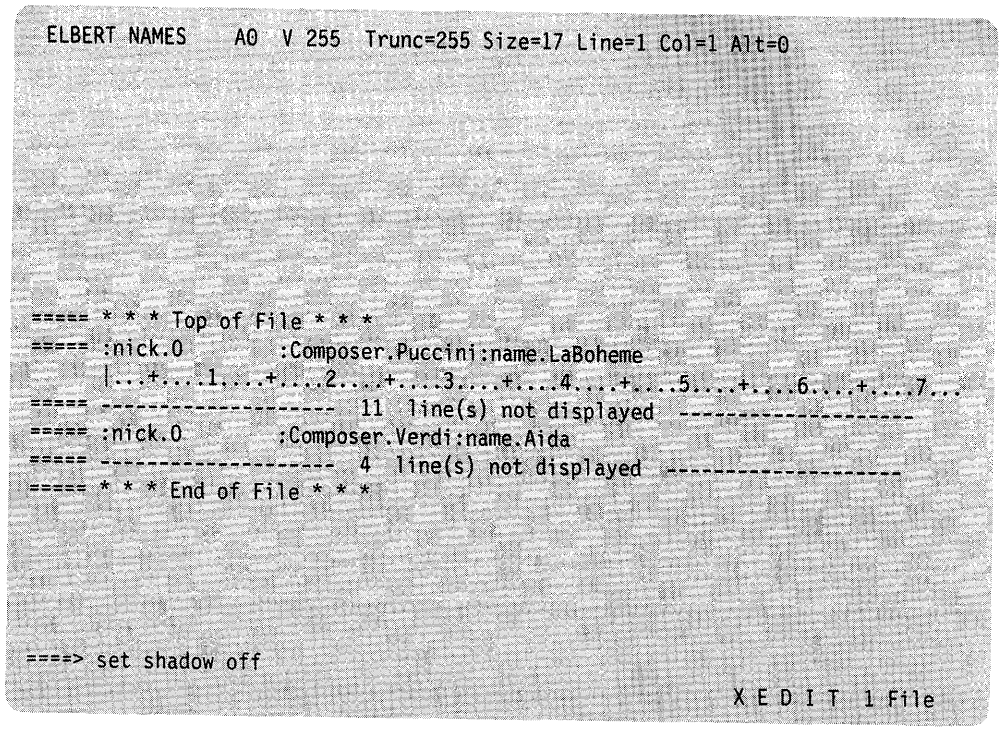
Resulting file displaying all classical records



```
 ELBERT NAMES    A0  V 255  Trunc=255 Size=17 Line=4 Col=1 Alt=0








===== * * * Top of File * * *
===== -------------------- 3  line(s) not displayed --------------------
===== :nick.C       :Composer.Grieg:name.Peer Gynt Suites
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== -------------------- 2  line(s) not displayed --------------------
===== :nick.C       :Composer.Ravel:name.Piano Concerto in G Major
===== -------------------- 2  line(s) not displayed --------------------
===== :nick.C       :Composer.Offenbach:name.Les Bavards
===== -------------------- 5  line(s) not displayed --------------------
===== :nick.C       :Composer.Mozart:name.Eine kleine Nachtmusik
===== -------------------- 1  line(s) not displayed --------------------
===== * * * End of File * * *

====> all/nick.O/
                                                      X E D I T  1 File
```

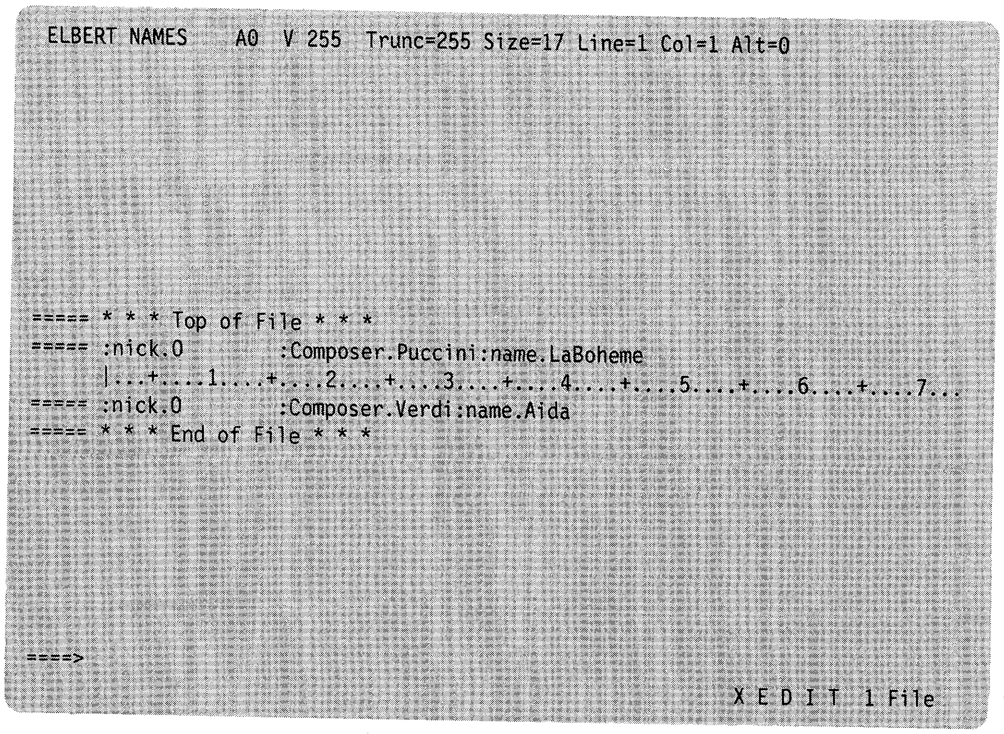
Using the ALL macro to select and display all operas in Elbert's collection

Figure 2 (Part 2 of 3). The ALL Macro



```
 ELBERT NAMES    A0  V 255  Trunc=255 Size=17 Line=1 Col=1 Alt=0






===== * * * Top of File * * *
===== :nick.0       :Composer.Puccini:name.LaBoheme
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== -------------------- 11  line(s) not displayed --------------------
===== :nick.0       :Composer.Verdi:name.Aida
===== -------------------- 4  line(s) not displayed --------------------
===== * * * End of File * * *




====> set shadow off
                                                                 X E D I T  1 File
```

Resulting file displaying all opera records



```
 ELBERT NAMES    A0  V 255  Trunc=255 Size=17 Line=1 Col=1 Alt=0






===== * * * Top of File * * *
===== :nick.0       :Composer.Puccini:name.LaBoheme
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== :nick.0       :Composer.Verdi:name.Aida
===== * * * End of File * * *






====>
                                                                 X E D I T  1 File
```

File, displaying all opera records, with shadow lines set off

Figure 2 (Part 3 of 3). The ALL Macro

ALTER (Macro)

Use the ALTER macro to change a single character to another character, one that may not be available on your terminal keyboard. The ALTER macro allows you to reference characters by their hexadecimal values.

Format

```
ALter   char 1  char 2 [ target  [ n  [ p ] ] ]
                                   *
                         1         1    1
                                   G
```

Operands

char1
: is the character to be altered. It may be specified either as a single character or in hexadecimal notation (00 through FF).

char2
: specifies the character to which char1 is to be altered. It may be specified either as a single character or in hexadecimal notation.

target
: defines the number of lines to be searched for char1. The search for char1 starts with the current line and continues up to, but does not include, the target line. If you specify an asterisk (*), the search continues to the end of the file (or the end of the range – see SET RANGE). If you omit target, only the current line is altered.

  A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide*.

n
: is the number of occurrences of char1 to be altered in each line examined. If you specify an asterisk (*), all occurrences of char1 are altered. If you omit n, only one occurrence of char1 in each line is altered. For compatibility with the CMS editor (EDIT), the G (Global) operand may be specified, but only when target is specified as a number.

p
: specifies the relative number of the first occurrence of char1 to be altered in each line examined. If you omit p, the alteration starts with the first occurrence of char1 in a line.

Responses

The column pointer remains unchanged.

If SET STAY OFF is in effect (the default), the last line examined becomes the new current line.

If SET STAY ON is in effect, the line pointer remains unchanged.

When verification is on, every line that is changed is displayed.

On a display terminal, when verification is off and a change has been made, the following message is displayed:

```
517I nn occurrence(s) changed on nn line(s)
```

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
1	TOF or EOF reached
2	Target line not found
4	No change occurred
5	Missing or invalid operand or invalid number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

You can use ALTER to change a special character to a backspace character, in order to produce compound characters on printed output.

Current Line:

```
===== Please underline T$_H$_I$_S$_
```

`alter $ 16 1 *` (alter $ to X'16' each time it appears in current line)

```
===== Please underline T _H _I _S _

When printed, the line will look like this:

Please underline THIS
```

BACKWARD

Use the BACKWARD subcommand to scroll backward toward the beginning of a file for a specified number of screen displays.

Format

BAckward	[*n* | * | <u>1</u>]

Operand

n

is the number of screen displays you want to scroll backward. If you specify an asterisk (*), the line pointer moves to the "Top of File" line. If you omit n, the screen scrolls back one display.

Usage Notes

1. The editor assigns the BACKWARD subcommand to the PF7 key.
2. If you issue a BACKWARD subcommand when the current line is the "Top of File" line, the editor "wraps around" the file, making the last line of the file the new current line.
3. Issuing `BACKWARD 0` from anywhere in the file makes the last line of the file the new current line.

Messages

520E	Invalid operand: *operand* [RC=5]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
543E	Invalid number: *number* [RC=5]

Return Codes

0	Normal
1	Top of File reached (subsequent BACKWARD restarts at end of file)
3	Terminal is not a display
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

BOTTOM

Use the BOTTOM subcommand to make the last line of the file or of the range (see the SET RANGE subcommand) the new current line.

Format

Bottom	

Usage Notes

1. One way to begin entering new lines at the end of a file is to issue the BOTTOM subcommand followed by the INPUT subcommand.
2. While the BOTTOM subcommand moves the line pointer to the last file line, a LOCATE * subcommand moves it to the null "End of File" (or "End of Range") line that follows the last line of the file. Use LOCATE * instead of BOTTOM if you intend to follow with an upward search for the *last* occurrence of a string within a file, (because the upward search starts with the line preceding the current line).

Message

520E Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

CANCEL (Macro)

Use the CANCEL macro when editing multiple files to terminate the editing session for all of the files. The CANCEL macro is equivalent to entering a QUIT subcommand for each file.

Format

CANCEL	

Usage Note

1. The QUIT that is issued against all the files is either *protected* or *unprotected*, depending on the defined synonyms (see the QUIT subcommand). If the QUIT subcommand has been defined to perform a protected QUIT, then CANCEL will quit all unmodified files but will issue a warning message for each modified file, leaving the user in edit mode. If all the files being cancelled were unmodified, the CANCEL macro causes an immediate exit from the editor.

Response

(if protected QUIT is defined):

```
File has been changed; use QQUIT to quit anyway
```

Messages

520E	Invalid operand: *operand* [RC=5]
577E	File has been changed; type QQUIT to quit anyway [RC=12]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	File has been changed (protected QUIT)

CAPPEND (Macro)

Use the CAPPEND macro to append specified text to the end of the current line.

Format

CAppend	[*text*]

Operand

text
is the text to be appended to the end of the current line. If no text is specified, the column pointer is placed under the first trailing blank.

Usage Notes

1. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.
2. The text operand starts with the first character following the blank delimiter after the subcommand name.
3. Column pointer movement is affected by the current zone setting. See the description of the SET ZONE subcommand for more complete information.

Response

The column pointer is placed under the first character of the appended text.

Messages

503E	{Truncated|Spilled} [RC=3]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
3	Truncated or spilled
4	No lines changed
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== It is an ancient mariner,
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

```
cappend and he stoppeth one of three.
```

(one blank between subcommand name and operand)

```
===== It is an ancient mariner,and he stoppeth one of three.
      <...+....1....+....2....+|...3....+....4....+....5....+....6....+....7...
```

Current Line:

```
===== It is an ancient mariner,
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

```
cappend  and he stoppeth one of three.
```

(two blanks between subcommand name and operand)

```
===== It is an ancient mariner, and he stoppeth one of three.
      <...+....1....+....2....+|...3....+....4....+....5....+....6....+....7...
```

CDELETE

Use the CDELETE subcommand to delete one or more characters from the current line, starting at the column pointer.

Format

CDelete	[*column-target* |1]

Operand

column-target
defines the number of characters to be deleted. Deletion starts at the column pointer and continues up to, but does not include, the column-target.

For a complete description of column-targets, refer to the CLOCATE subcommand.

Usage Notes

1. Use the CLOCATE subcommand to move the column pointer to the column you want deletion to begin at.
2. As with all column-targets, the following SET options have an effect on the column-target search:

SET ARBCHAR	SET SPAN
SET CASE	SET VARBLANK
SET ETARBCH.	

3. When SET STREAM OFF has been issued, only the current line is searched for the string to be deleted. When SET STREAM ON has been issued, the editor searches for the string up to the end of the file (or range) or to the top of the file (or range) if the search is in a backward direction. In this case, several lines may be deleted.

Response

If SET STREAM ON is in effect, and more than one line was deleted, the following message is displayed:

```
501I  nn line(s) deleted
```

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]
585E	No line(s) changed [RC=4]
700E	Logical AND operator & not valid for column targets [RC=5]

Return Codes

0	Normal
2	Target not found
4	No line(s) changed
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== There are now more than 3,000 languages in the world.

      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`cl  :11` (move the column pointer)
`cd  :15` (delete characters from the column pointer to column 15)

```
===== There are more than 3,000 languages in the world.
      <...+....1|...+....2....+....3....+....4....+....5....+....6....+....7...
```

Current Line:

```
===== A dialect is considered a language if it is used in newspapers.

      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`cl  :43`
`cdelete  -6` (delete characters in current column and 5 preceding ones)

```
===== A dialect is considered a language if used in newspapers.

      <...+....1....+....2....+....3....+....4..|.+....5....+....6....+....7...
```

Current Line:

```
===== Russian is spoken by 190 million people.

      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`cdelete /spoken/` (delete characters from the column pointer to the first character of the string)

```
===== spoken by 190 million people.

      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

Current Line:

```
===== Russian is spoken by 190 million people.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`cl :39`
`cd -/190/` (delete characters from column pointer up to the first character of the string)

```
===== Russian is spoken by 1.
      <...+....1....+....2....+....3....+...|4....+....5....+....6....+....7...
```

CFIRST

Use the CFIRST subcommand to move the column pointer to the beginning of the zone (see SET ZONE).

Format

CFirst	

Usage Note

After subcommands that move the column pointer have been executed, use the CFIRST subcommand to reset the column pointer to the left zone.

Message

520E Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

(Note the position of the column pointer.)

```
===== The automobile heater was invented by a woman from Brooklyn.
      <...+....1....+....2....+....3....+....4|...+....5....+....6....+....7...

cfirst

===== The automobile heater was invented by a woman from Brooklyn.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

CHANGE

Use the CHANGE subcommand to change a specified group of characters to another group of characters of the same or a different length. You can use the CHANGE subcommand to change more than one line at a time.

Format

```
Change  │ / string 1 [/ string 2 / [target [p [q]]]]
        │                           1      *  1
        │                                  1
```

Operands

/ (diagonal)
: signifies any delimiting character that does not appear in the character strings involved in the change.

string1
: is a group of characters to be changed (old data).

string2
: is the group of characters that is to replace string1 (new data). If string2 is omitted, it is assumed to be a null string. The trailing delimiter may be necessary in certain circumstances. For example, if string2 has trailing blanks, the trailing delimiter should be used to indicate where the string ends.

target
: defines the number of lines to be changed. Lines are changed starting with the current line, up to but not including the target line. If you specify an asterisk (*), lines are changed until the end of the file (or the end of the range). If you omit target, only the current line is changed.

  A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*

p
: is the number of occurrences of string1 to be changed in each line. If you specify *, string1 is changed every time it appears in a line. If you omit p, string1 is changed only once.

q
: is the relative number of the first occurrence of string1 to be changed in each line. If you omit q, the change starts with the first occurrence of string1 in each line.

Usage Notes

1. The first nonblank character following the CHANGE subcommand is considered to be the delimiter.

   For example:

   `change .VM/370.CMS.` changes `VM/370` to `CMS`

2. If string2 is longer than string1, and if SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. If a line is spilled, no additional changes will be made on that line.

   If string2 is shorter than string1, characters are shifted left (from the truncation column), and the line is padded with blanks (up to the truncation column).

3. If string1 is represented as a null string, string2 is inserted in the line, starting at the beginning of the zone, which may or may not be column 1.

4. Using CHANGE with SET ARBCHAR ON:

   ```
   set arbchar on .
   change /(.)/'.'/
   ```

   The expression that was in parentheses is now enclosed by quotation marks.

   ```
   change /(.)//
   ```

   String2 is represented as a null string. As a result, the expression in parentheses (and the parentheses) is deleted.

   ```
   set arbchar on $
   change /y$/y/
   ```

   results in all characters following y being deleted, ending at zone 2 (not truncation column). Characters from zone 2 to the truncation column are shifted left and the line is padded with blanks (up to the truncation column).

   For additional examples of using CHANGE with SET ARBCHAR, refer to the "Examples" section below and to the SET ARBCHAR subcommand description.

   To use CHANGE with SET ETARBCH see Appendix F, "Using Double-Byte Character Sets" on page 419.

5. Using SET CASE and CHANGE:

   String1 should be typed exactly as it appears in the file in order for it to be changed (with SET CASE MIXED).

6. Using SET STAY and CHANGE:

   If you specify that a change is to occur on multiple lines and the change occurs, the current line pointer will be:

   a. Unchanged, if SET STAY ON has been entered
   b. Moved to the last line scanned, if SET STAY OFF is in effect (the default).

7. Using SET ZONE and CHANGE:

   The search for string1 occurs only between the left and right zones. However, characters are shifted left or the line is padded with blanks from the right zone up to the truncation column, as explained in usage note 2.

```
set arbchar on $
change /$/xy/
```

results in all characters from zone 1 through zone 2 (not the truncation column) being replaced by characters xy. Characters from zone 2 to the truncation column are shifted left and the line is padded with blanks (up to the truncation column).

8. The search for string1 is not affected by the setting of SET SPAN or SET VARBLANK.
9. The CHANGE subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Responses

On a typewriter terminal, when verification is on, every line that is changed is displayed.

On a display terminal, when verification is off and a change is made, one of the following messages is displayed:

```
518E    nn occurrence(s) changed on nn line(s);
        nn line(s) {truncated|spilled} [RC=3]
517I    nn occurrence(s) changed on nn line(s)
```

Messages

503E	{Truncated|Spilled} [RC = 3]
511E	String2 contains more arbitrary characters than string1 [RC = 5]
520E	Invalid operand: *operand* [RC = 5]
543E	Invalid number: *number* [RC = 5]
545E	Missing operand(s) [RC = 5]
546E	Target not found [RC = 2]
585E	No line(s) changed [RC = 4]

Return Codes

0	Normal
1	TOF or EOF reached during change
2	Target line not found
3	Truncation or spill occurred during the change
4	No change occurred. (string1 has not been found)
5	Invalid or missing operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== A rose is a rose is a rose.
```

`change/rose/daisy/` (change first occurrence in the current line)

```
===== A daisy is a rose is a rose.
```

`change/rose/daisy/ 1 *` (change all occurrences in the current line)

```
===== A daisy is a daisy is a daisy.
```

The following subcommand would change every occurrence of "rose" to "daisy" in every line of the file, beginning with the current line.

```
change/rose/daisy/ * *
```

Current Line:

```
===== James Bernard is my favorite artist.
```

```
change//Mr. / (insert "Mr. " in column 1)
```

```
===== Mr. James Bernard is my favorite artist.
```

Using CHANGE with SET ARBCHAR ON:

```
===== Lewis Carroll wrote (The Walrus and the Carpenter).
```

`change/($)/"$"/` (change parentheses to quotation marks)

```
===== Lewis Carroll wrote "The Walrus and the Carpenter".
```

```
===== Robert Browning wrote (among other things) "My Last Duchess".
```

`change  / ($)//` (string2 is a null string)

```
===== Robert Browning wrote "My Last Duchess".
```

CINSERT

Use the CINSERT subcommand to insert text in the current line starting at the column pointer. As a result, the data is shifted to the right.

Format

CInsert	*text*

Operand

text
is the group of characters to be inserted starting at the column pointer.

Usage Notes

1. You can insert blanks with the CINSERT subcommand. The operand must contain the number of blanks you want to insert. (You cannot enter the CINSERT subcommand without an operand.)
2. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been entered, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.
3. Use the CLOCATE subcommand to move the column pointer to the desired location.
4. If the column pointer is at Zone1 − 1 (TOL) or Zone2 + 1 (EOL), no characters will be inserted.

Response

The column pointer remains unchanged.

Messages

503E	{Truncated|Spilled} [RC=3]
545E	Missing operand(s) [RC=5]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
3	Truncated or spilled
4	No line(s) changed
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

Current Line:

```
===== Mount Everest is high.
      |...+....1....+....2....+....3....+....4....+....5....+....6....
```

`cl /high/` (move the column pointer)
`ci exactly 29,000 feet` (one blank entered after "feet" for spacing)

```
===== Mount Everest is exactly 29,000 feet high.
      <...+....1....+..|.2....+....3....+....4....+....5....+....6....
```

CLAST

Use the CLAST subcommand to move the column pointer to the end of the zone (see SET ZONE).

Format

CLAst	

Message

520E Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples:

```
set zone 1 20
```

Current Line:

```
===== Harvey Kennedy invented the shoelace and made $2.5 million.
      |...+....1....+....>....+....3....+....4....+....5....+....6....+....7...
```

```
clast
```

```
===== Harvey Kennedy invented the shoelace and made $2.5 million.
      <...+....1....+....|....+....3....+....4....+....5....+....6....+....7...
```

CLOCATE

Use the CLOCATE subcommand to scan the file for a specified column-target, and to move the column pointer to the target, if located. The search begins with the column following (or preceding) the column pointer in the current line. The CLOCATE subcommand is used to find successively *all* occurrences of a character string and to move the column pointer if the string is found.

Format

CLocate	*column-target*

Operand

column-target
can be specified as an absolute column number, a relative column number, a string expression, or a complex string expression. A complete description of column-targets follows, under "Usage Notes."

Usage Notes — Column-targets

A column-target is a specialized operand used only in the CLOCATE and CDELETE subcommands.

It is not to be confused with the *target* operand used in many other XEDIT subcommands and macros. That kind of target is actually a line target. When a line target is found, the line pointer is moved, but the column pointer is not moved. If a line target is expressed as a string, only the first occurrence of the string will be located in each line, regardless of how many times the string appears in a line. For example, if a line contains more than one occurrence of the string "ABC" and you issue a LOCATE subcommand for it, the line pointer moves to that line. If the same LOCATE subcommand is repeated, the line pointer would move to the *next* line containing "ABC".

On the other hand, when a *column-target* is expressed as a string and that string is found, the column pointer is moved to the first character of the string. If the string appears in this line more than once, repeated CLOCATE subcommands move the column pointer to the first character of the string for each occurrence located. In addition, if SET STREAM ON is in effect (the default), the line pointer is also moved, making it possible to locate all occurrences of the string throughout the file (by repeated executions of the CLOCATE subcommand).

The CLOCATE subcommand is used to move the column pointer to a specified column or to locate a specified string; the column pointer is moved if the string is found.

A column-target may be expressed in the following ways:

1. An *absolute column number* is used to move the column pointer. It is expressed as a colon followed by an integer. For example:

   ```
   clocate :2
   ```

   moves the column pointer to column 2 of the current line.

Use this form of the CLOCATE subcommand when you plan to enter subsequently a subcommand that starts its operation at the column pointer, for example, JOIN COLUMN.

2. A *relative column number* is used to move the column pointer. It is expressed as an integer and may be preceded by a plus (+) or minus (−) sign, which indicates a right (+) or left (−) move. If the sign is omitted, a plus (+) is assumed.

   For example:

   ```
   clocate +2
   clocate  2
   ```

   both move the column pointer two columns to the right of its current position. A relative column number may also be specified as an asterisk (*), which means one column to the left of the left zone (−*) or one column to the right of the right zone (+* or *). By using CLOCATE −* or * first, you can then use CLOCATE to find a string, even if it is in the first or last column of the zone. You can determine when the column pointer has reached either Zone1 − 1 or Zone2 + 1 by using the TOL (Top of Line) or EOL (End of Line) operands on the QUERY or EXTRACT subcommand.

3. A *string expression* defines a group of characters to be located, starting with the column immediately following (or preceding, depending on the direction of the search) the current column.

   If SET HEX ON has been issued, a string may be specified in hexadecimal notation, and the EBCDIC equivalent will be searched for.

   The format of a string expression is:

```
[+|-][¬]/string1[/|[¬]/string2/]...
  1  2    3    4
```

[1] Right (+) or left (−) search (right is the default)

[2] "NOT" symbol (locate something that is not the specified string)

[3] character (or hexadecimal) string. The trailing delimiter may be necessary in certain circumstances. For example, if string1 has trailing blanks, the trailing delimiter should be used to indicate where the string ends.

[4] "OR" symbol (vertical bar) (locate any of the strings, separated by OR symbols, starting with the first string specified).

Examples

`clocate /horse/`
: searches the file for the first occurrence of "horse," starting at the first character following the column pointer.

`clocate -/horse/`
: searches *backward* for the first occurrence of "horse," starting at the first character preceding the column pointer.

`clocate ¬/horse/`
: searches for the first occurrence that is *not* "horse," starting at the first character following the column pointer.

`clocate /horse/|/buggy/`

searches each line for "horse," even if "buggy" occurs before "horse" in the line. If "horse" is not found, then searches the line for "buggy," and if neither is found, the search is repeated until the end (or top) of the file.

`clocate //`

advances the column pointer one column.

4. A *complex string expression* can have the same format as a string expression (see above), but any string can be expressed as a "complex string," which is defined as a string associated with one or more of the following SET subcommand options:

   ARBCHAR
   CASE
   ETARBCH
   SPAN
   VARBLANK.

   See the LOCATE subcommand for a description of these options.

5. SET STREAM is meaningful only with column-targets.

   `stream on`

   specifies that each line is searched for a string that matches the column-target starting with the character following (or preceding) the column pointer on the current line and continues to the end (or top) of the file (or range) until a match is found. STREAM ON is the default setting.

   If SET WRAP is also ON (OFF is the default) and the editor "wraps around" the file (see SET WRAP) if the top of file (or range) or end of file (or range) is reached during a CLOCATE, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```

   `stream off`

   specifies that the search is confined to the current line (or zones within the line).

6. The CLOCATE subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]
546E	Target not found [RC = 2]
592W	Wrapped ....
700E	Logical AND operator & not valid for column targets [RC = 5]

Return Codes

0	Normal
1	Out of zone definition during execution
2	Target not found (character pointer stays where it was)
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== John Keats studied medicine and practiced as an apothecary.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`clocate :6` (absolute column number)

```
===== John Keats studied medicine and practiced as an apothecary.
      <...+|...1....+....2....+....3....+....4....+....5....+....6....+....7...
```

Current Line:

```
===== James Joyce was a school teacher in Dublin.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`clocate +6` (relative column number)

```
===== James Joyce was a school teacher in Dublin.
      <...+.|..1....+....2....+....3....+....4....+....5....+....6....+....7...
```

Current Line:

```
===== Herman Melville worked as a customs inspector in N.Y.C.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

```
clocate /customs/
```

```
===== Herman Melville worked as a customs inspector in N.Y.C.
      <...+....1....+....2....+...|3....+....4....+....5....+....6....+....7...
```

Current Line:

```
===== Charles Dickens served as a law clerk and was a reporter.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`clocate /reporter/|/clerk/` (locate "reporter" or "clerk")

```
===== Charles Dickens served as a law clerk and was a reporter.
      <...+....1....+....2....+....3....+....4....+...|5....+....6....+....7...
```

CMS

Use the CMS subcommand to have the editor transmit a command to CMS for execution, or to have the editor enter CMS subset mode.

Format

CMS	[*commandline*]

Operand

commandline
is any CMS command. If no command is specified, the editor enters CMS subset mode.

Usage Notes

1. If you enter the CMS subcommand *without* an operand, you enter CMS subset mode. For a description of CMS subset mode, please refer to the *VM/SP CMS User's Guide*. You must use the CMS subset command RETURN to return to edit mode.
2. If you try to execute a CMS command that terminates abnormally, changes made during your editing session might be lost. You should save the input you have entered before you use the CMS subcommand.
3. Any CMS command should be prefaced with 'CMS' to prevent XEDIT from decoding the subcommand. This should be done to prevent cases when CMS commands may be interpreted as XEDIT subcommands.
4. Some CMS commands issue either informational messages or error messages to the XEDIT environment. While in CMS subset mode, you will not see these messages until you return to XEDIT unless MSGLINE is set OFF or the maximum number of message lines has been exceeded. However, you will see the return code associated with the message as part of the CMS ready message.

Responses

When you enter the CMS subcommand without an operand, the following message is displayed:

```
CMS subset
```

This indicates you are in CMS subset mode. If full-screen CMS is ON (via the SET FULLSCREEN ON command), the message will appear in the CMS window connected to the virtual screen. Also, the STATUS window will be popped. If full-screen CMS is OFF, the editor clears the screen before issuing the CMS SUBSET message. To return to the XEDIT environment when full-screen CMS is ON **or** OFF, use the CMS RETURN command.

When you enter a CMS subcommand with an operand and the CMS command does not write on the screen, the file image remains on the screen.

If full-screen CMS is ON and the CMS command displays text, then the text will appear in the CMSOUT window. (Text will appear in the CMSOUT window by default; you can route the text to another window by using the CMS ROUTE command, explained in the *VM/SP CMS Command Reference*). To see all of the text in the virtual screen, you can use the CMS SCROLL command. The screen will be cleared automatically when you scroll to the bottom of the virtual screen. An alternative way to clear the screen is to issue the CMS DROP WINDOW command. If you delete the CMSOUT window, you won't see messages that are passed to CMS.

If full-screen CMS is OFF, the text replaces the file image on the screen and the terminal is placed in a "MORE..." (waiting) status. To get the file image back on the screen, either press the CLEAR key or wait for one minute.

If the return code from the CMS command is a negative number other than -2, the message 'Unknown CP/CMS command' will be issued. For -2, the message 'Invalid subset command' is issued.

Messages

512E	Invalid subset command [RC=-2]
513E	Unknown CP/CMS command [RC=-3]
514E	Return code *nn* from *command* [RC=*nn*]

Return Codes

-2	Invalid subset command
-3	Unknown CMS/CP command
0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
-nn	Unknown CP/CMS command
nn	The return code of the CMS command after RETURN to XEDIT environment

CMSG

Use the CMSG subcommand to display a message in the command line of the terminal. The CMSG subcommand is intended to be issued from a macro.

Format

CMSG	[*text*]

Operand

text
is the text of the message to be displayed. If no text is specified, the command line is reset to a blank line.

Usage Notes

1. If it is issued to a typewriter terminal, the CMSG subcommand has no effect.
2. If it is issued when the cmdline is off, the CMSG subcommand has no effect.
3. If it is issued when the stack is not empty, the CMSG subcommand has no effect.

Notes for Macro Writers

When issued from a macro, CMSG can be used to redisplay input that the user has entered incorrectly, so that the input can be corrected and reentered.

Response

The message is displayed in the command line.

Return Codes

0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

COMMAND

Use the COMMAND subcommand to have the editor execute an XEDIT subcommand without first checking for identically-named synonyms and macros.

Format

COMMAND	[*commandline*]

Operand

commandline
is any XEDIT subcommand name and its operands, except for ? and &.

Usage Notes

1. The editor executes the specified subcommand even if the SET SYNONYM ON or SET MACRO ON subcommands have been issued. In other words, the editor does not check to see if a synonym or macro with the same name exists before it executes the specified subcommand.
2. If SET IMPCMSCP ON is in effect, any commands not recognized by the editor will be transmitted to CMS or to CP.

Response

The response, if any, from the executed subcommand is displayed.

Messages

542E No such subcommand: *name* [RC=-1]

Error messages from the executed subcommand (if any) are displayed.

Return Codes

nn	Return code of the subcommand specified as operand
-1	No such subcommand
0	Normal
6	Subcommand rejected in the profile due to LOAD error or QUIT subcommand has been issued in a macro called from the last file in the ring

COMPRESS

Use the COMPRESS subcommand to prepare one or more file lines, starting with the current line, for automatic repositioning of data according to new tab column settings. Use the SET TABS subcommand to define new tab column settings.

Format

COMPress	[*target* <u>1</u>]

Operands

target
defines the number of lines to be compressed, starting with the current line, up to, but not including, the target line. If you specify an asterisk (*), the rest of the file is compressed. If you omit target, only the current line is compressed.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*

Usage Notes

1. After you compress a line, you can use the SET TABS and EXPAND subcommands to reposition the data in the new tab columns.

   Using this sequence of commands:

   set tabs . . .
   compress . . .
   set tabs . . .
   expand . . .

   you can realign an entire table.

2. For COMPRESS to work properly, lines containing backspace characters (X'16') must have been entered into the file with SET IMAGE CANON; that is, compound characters ordered with backspaces arranged singly between the characters that overlay each other.

3. Lines are compressed according to the current SET TABS setting. The COMPRESS subcommand removes all contiguous blank characters (or whatever character is defined as a filler character by the SET FILLER subcommand) that immediately precede the current tab columns. It replaces each group of blank (or filler) characters with a tab character (X'05').

Responses

1. If you specify that a compress is to occur on multiple lines and the compress occurs, the current line pointer will be:

   a. Unchanged, if SET STAY ON has been entered
   b. Moved to the last line compressed, if SET STAY OFF is in effect (the default).

2. The data in a compressed line shifts left, reflecting the removal of blanks.

Messages

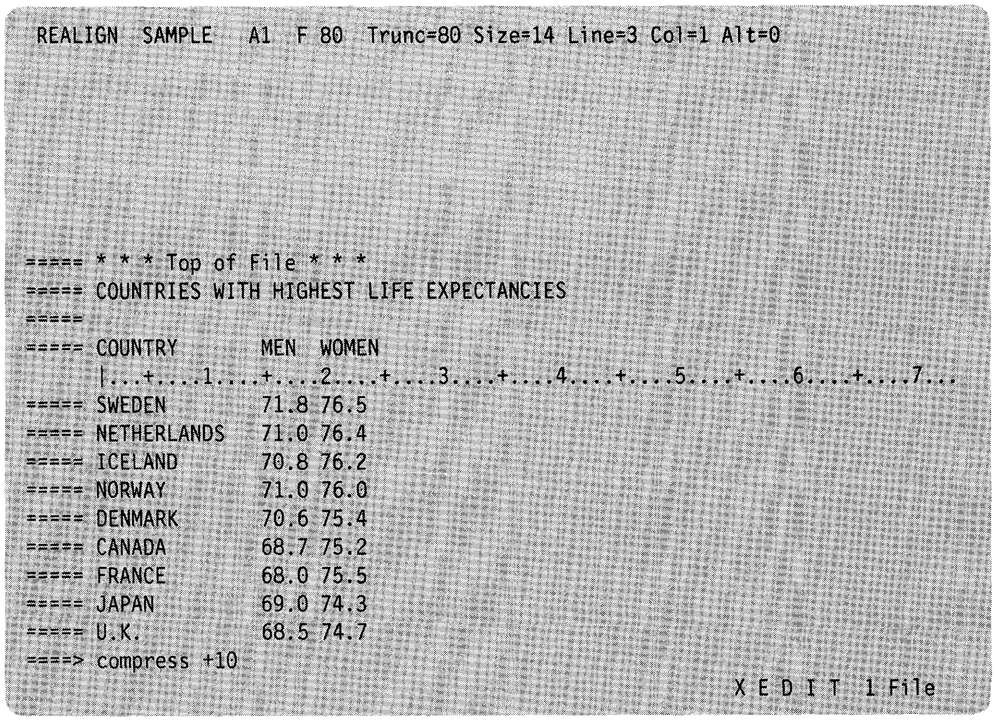
520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]
581E	Subcommand is not valid in extended mode [RC=3]
585E	No line(s) changed [RC=1 or 4]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
3	Subcommand is not valid in extended mode
4	No line(s) changed
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

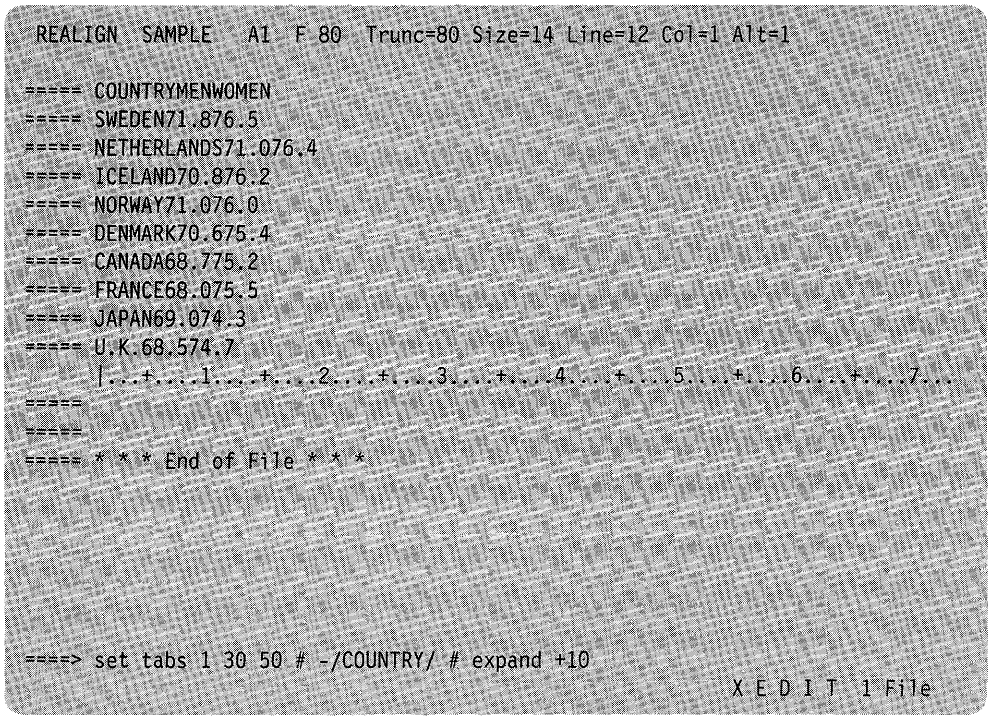
Examples

Figure 3, Figure 4, and Figure 5 are examples of using the COMPRESS and EXPAND subcommands to realign data in a table.



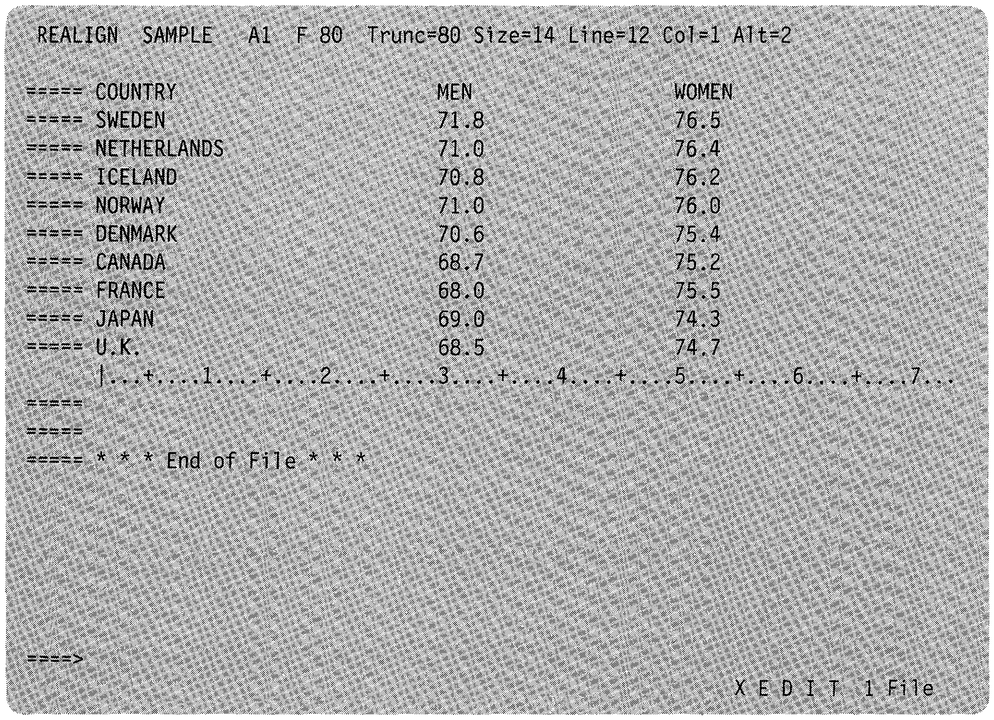
```
 REALIGN  SAMPLE   A1  F 80  Trunc=80 Size=14 Line=3 Col=1 Alt=0







===== * * * Top of File * * *
===== COUNTRIES WITH HIGHEST LIFE EXPECTANCIES
=====
===== COUNTRY        MEN  WOMEN
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== SWEDEN        71.8 76.5
===== NETHERLANDS   71.0 76.4
===== ICELAND       70.8 76.2
===== NORWAY        71.0 76.0
===== DENMARK       70.6 75.4
===== CANADA        68.7 75.2
===== FRANCE        68.0 75.5
===== JAPAN         69.0 74.3
===== U.K.          68.5 74.7
====> compress +10
                                                            X E D I T  1 File
```

Figure 3. Realigning a Table (Current tab setting is 1 15 20) — COMPRESS



```
 REALIGN  SAMPLE   A1  F 80  Trunc=80 Size=14 Line=12 Col=1 Alt=1

===== COUNTRYMENWOMEN
===== SWEDEN71.876.5
===== NETHERLANDS71.076.4
===== ICELAND70.876.2
===== NORWAY71.076.0
===== DENMARK70.675.4
===== CANADA68.775.2
===== FRANCE68.075.5
===== JAPAN69.074.3
===== U.K.68.574.7
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====
===== * * * End of File * * *







====> set tabs 1 30 50 # -/COUNTRY/ # expand +10
                                                             X E D I T  1 File
```

Figure 4. Realigning a Table — Set New Tabs and EXPAND



```
 REALIGN  SAMPLE   A1  F 80  Trunc=80 Size=14 Line=12 Col=1 Alt=2

===== COUNTRY                      MEN                 WOMEN
===== SWEDEN                       71.8                76.5
===== NETHERLANDS                  71.0                76.4
===== ICELAND                      70.8                76.2
===== NORWAY                       71.0                76.0
===== DENMARK                      70.6                75.4
===== CANADA                       68.7                75.2
===== FRANCE                       68.0                75.5
===== JAPAN                        69.0                74.3
===== U.K.                         68.5                74.7
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====
===== * * * End of File * * *







====>
                                                             X E D I T  1 File
```

Figure 5. Realigning a Table — Realigned Table

COPY

Use the COPY subcommand to copy one or more lines, beginning with the current line, at a specified location in the file.

Format

COpy	*target* **1** *target* **2**

Operands

target1
defines the number of lines to be copied. Lines are copied starting with the current line, up to but not including target1.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*

target2
defines the line after which the data is to be copied.

Responses

The last line that was copied becomes the new current line.

The editor displays the following message:

```
506I nn lines copied
```

Messages

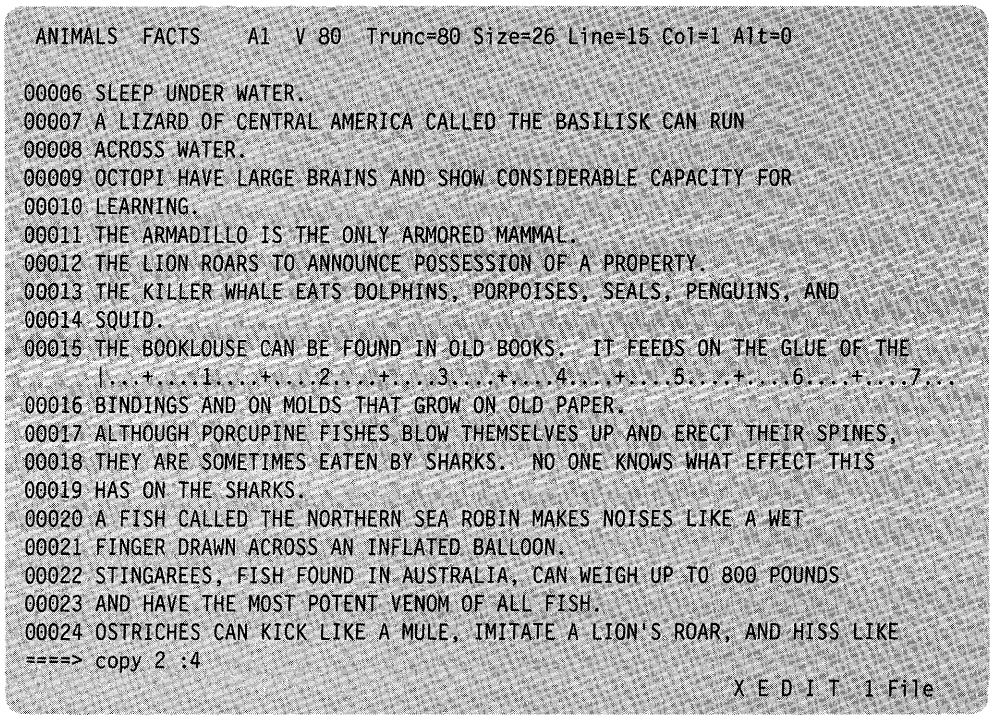
520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
557S	No more storage to insert lines [RC=4]

Return Codes

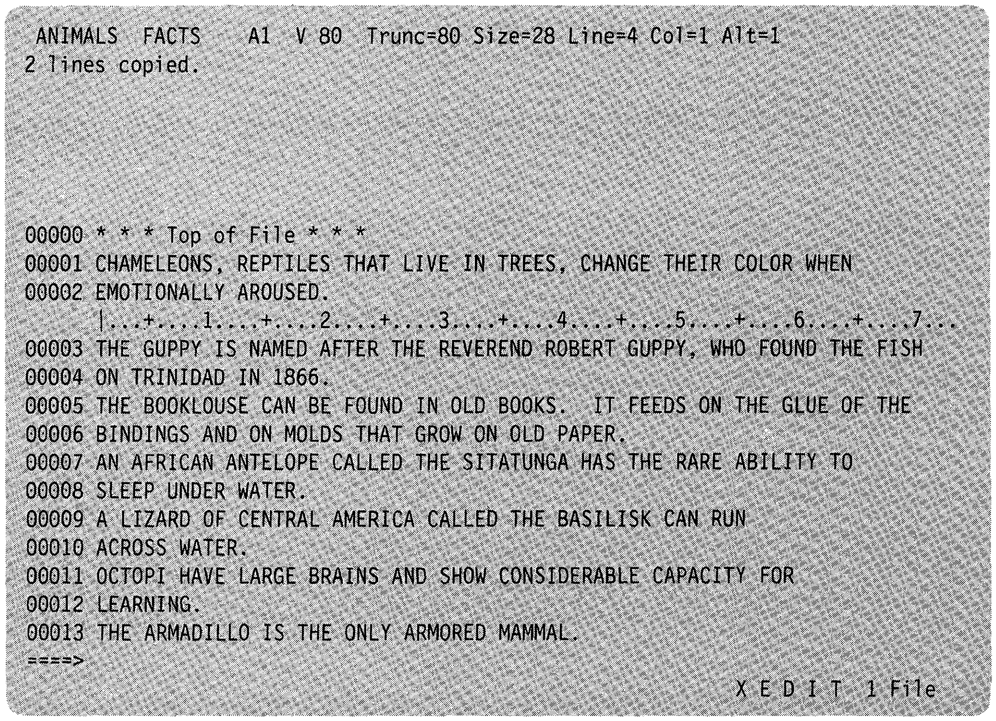
0	Normal
2	Target line not found
4	No more storage available
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 6 is a before-and-after example of the COPY subcommand.



```
 ANIMALS  FACTS    A1  V 80  Trunc=80 Size=26 Line=15 Col=1 Alt=0

00006 SLEEP UNDER WATER.
00007 A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
00008 ACROSS WATER.
00009 OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
00010 LEARNING.
00011 THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
00012 THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
00013 THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
00014 SQUID.
00015 THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS.  IT FEEDS ON THE GLUE OF THE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00016 BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
00017 ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
00018 THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
00019 HAS ON THE SHARKS.
00020 A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
00021 FINGER DRAWN ACROSS AN INFLATED BALLOON.
00022 STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
00023 AND HAVE THE MOST POTENT VENOM OF ALL FISH.
00024 OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISS LIKE
====> copy 2 :4
                                                            X E D I T  1 File
```



```
 ANIMALS  FACTS    A1  V 80  Trunc=80 Size=28 Line=4 Col=1 Alt=1
 2 lines copied.





00000 * * * Top of File * * *
00001 CHAMELEONS, REPTILES THAT LIVE IN TREES, CHANGE THEIR COLOR WHEN
00002 EMOTIONALLY AROUSED.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00003 THE GUPPY IS NAMED AFTER THE REVEREND ROBERT GUPPY, WHO FOUND THE FISH
00004 ON TRINIDAD IN 1866.
00005 THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS.  IT FEEDS ON THE GLUE OF THE
00006 BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
00007 AN AFRICAN ANTELOPE CALLED THE SITATUNGA HAS THE RARE ABILITY TO
00008 SLEEP UNDER WATER.
00009 A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
00010 ACROSS WATER.
00011 OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
00012 LEARNING.
00013 THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
====>
                                                            X E D I T  1 File
```

Figure 6. The COPY Subcommand — Before and After

COUNT

Use the COUNT subcommand to display the number of times a specified character string appears in one or more lines, starting with the current line.

Format

COUnt	*/string* [*/target* 1]

Operands

string
is the character string to be counted.

target
defines the number of lines to be searched. The search begins with the current line and continues up to, but does not include, the target line. If you specify an asterisk (*), the search continues to the end of file (or the end of range—see SET RANGE). If you omit target, only the current line is searched.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*

Usage Notes

1. The count corresponds to the number of strings that would be changed by a CHANGE subcommand.
2. Arbitrary characters (see the SET ARBCHAR subcommand) can be specified in the */string/* operand.

   For example:

   ```
   set arbchar on $
   count /($)/ *
   ```

   will count all of the expressions enclosed in parentheses.
3. In a macro, you can use EXTRACT/LASTMSG/ to get the number of occurrences.
4. The *string* should be typed exactly as it appears in the file in order for it to be counted.

Responses

If you specify that a COUNT is to occur on multiple lines and the count occurs, the current line pointer will be:

a. Unchanged, if SET STAY ON has been entered
b. Moved to the last line scanned, if SET STAY OFF is in effect (the default).

The editor displays the number of times the string appears with the following message:

```
522I nn occurrences
```

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
5	Missing or invalid operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== A rose is a rose is a rose.
```

```
count/rose/
```

Response:

```
522I 3 occurrences
```

COVERLAY

Use the COVERLAY subcommand to replace selectively one or more characters in the current line with the corresponding characters in the text that is keyed in. Replacement starts at the column pointer.

Format

COVerlay	*text*

Operand

text
is a group of characters that replaces characters in corresponding positions in the current line.

Usage Notes

1. Blank characters in the text operand indicate that the corresponding characters in the current line are *not* to be overlaid.

   For example:

Current line:	`ABCDE`
COVERLAY subcommand:	`coverlay MN PQ`
Result:	`MNCPQ`

2. An underscore character (_) is used in the text operand to place a blank in the corresponding character position in the current line.

   Therefore, you cannot use this subcommand to place an underscore character in a line.

   Use the COVERLAY command carefully if a line contains underscored words or other compound characters.

3. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been entered, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.

4. If the column pointer is at Zone1 − 1 (TOL) or Zone2 + 1 (EOL), no characters will be overlaid.

Messages

503E	{Truncated|Spilled} [RC = 3]
545E	Missing operand(s) [RC = 5]
581E	Subcommand is not valid in extended mode [RC = 3]
585E	No line(s) changed [RC = 4]

Return Codes

0	Normal
3	Truncated or spilled, or subcommand is not valid in extended mode
4	No lines changed
5	Missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== Shall I compare thee to a summer's day?
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

```
cl /summer/  (move the column pointer)
cov winter   night?  (blanks are not overlaid)
```

```
===== Shall I compare thee to a winter's night?
      <...+....1....+....2....+.|..3....+....4....+....5....+....6....+....7...
```

CP

Use the CP subcommand to transmit commands to the VM/SP control program environment during an editing session.

Format

CP	[*commandline*]

Operand

commandline
is any CP command valid for your CP command privilege class. If you specify this operand, the editor transfers the command to CP and automatically returns to the editor environment. If you omit this operand, the editor enters the CP console function mode, where you can enter CP commands without preceding each command with CP. To return to the edit environment, enter the CP command BEGIN.

Usage Note

1. Any CP command should be prefaced with "CP" to prevent the editor from mistaking the CP command for an XEDIT subcommand.

Responses

When you issue the CP subcommand with an operand, and the CP command does not write on the screen, the file image remains on the screen.

If full-screen CMS is ON and the CP command displays text, text will appear in the CMSOUT window. Text will appear in the CMSOUT window by default; you can route the text to another window by using the CMS ROUTE command, explained in the *CMS Command Reference*. To see all of the text in the virtual screen, you can use the CMS SCROLL command. The screen will be cleared automatically when you scroll to the bottom of the virtual screen. An alternative way to clear the screen is to issue the CMS DROP WINDOW command. If you delete the CMSOUT window, you won't see messages that are passed to CMS.

If full-screen CMS is OFF, the text replaces the file image on the screen and the terminal is placed in a "MORE..." (waiting) status. To get the file image back on the screen, either press the CLEAR key or wait for one minute.

Message

513E	Unknown CP/CMS command [RC=-3]

Return Codes

0	Normal
−3	Unknown command
nn	Return code of the CP command
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

CREPLACE

Use the CREPLACE subcommand to replace one or more characters in the current line with a specified character or group of characters, starting at the column pointer.

Format

CReplace	*text*

Operand

text
specifies those characters that are to replace characters in the current line. This operand may contain all blanks, or it may contain imbedded blanks.

Usage Notes

1. Characters in the current line are replaced, one-for-one, with characters in the operand.
2. No shifting occurs, as with deletion or insertion of characters.
3. Use the CLOCATE subcommand to move the column pointer to the desired location.
4. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.
5. If the column pointer is at Zone1 − 1 (TOL) or Zone2 + 1 (EOL), no characters will be replaced.

Messages

503E	{Truncated|Spilled} [RC = 3]
545E	Missing operand(s) [RC = 5]
585E	No line(s) changed [RC = 4]

Return Codes

0	Normal
3	Truncated or spilled
4	No line(s) changed
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== Shall I compare thee to a summer's day?
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

`cl  /summer/` (move the column pointer)
`cr winter night?` (blanks may be imbedded)

```
===== Shall I compare thee to a winter night?
      <...+....1....+....2....+.|..3....+....4....+....5....+....6....+....7...
```

CURSOR

Use the CURSOR subcommand to move the cursor to a specified position and to assign a priority to the specified position. The cursor is positioned according to the highest assigned priority when all pending prefix subcommands and any macros are executed or when the screen is redisplayed.

Format

CURsor	**CMdline**	[*colno* 1] [**Priority** *n*]
	Column	[**Priority** *n*]
	File	*lineno* [*colno*] [**Priority** *n*]
	Home	[**Priority** *n*]
	Screen	*lineno* [*colno*] [**Priority** *n*]

Operands

CMdline
moves the cursor under the command line in the specified column.

Column
moves the cursor under the current line in the current column position.

File
moves the cursor relative to the beginning of the file.

Home
moves the cursor from the command line to the screen or vice versa, depending on its current position. If the cursor is on the screen, that is, any place but on the command line, CURSOR HOME remembers its location and moves it to the command line. If the cursor is on the command line, CURSOR HOME returns it to its previous location on the screen.

Screen
moves the cursor relative to the beginning of the logical screen.

lineno
specifies the file line (if used with the FILE operand) or the screen line (if used with the SCREEN operand) where the cursor is to be placed.

colno
specifies the column number where the cursor is to be placed. The location of colno varies according to the option with which it is specified:

If used with CMDLINE, colno places the cursor relative to the beginning of the command line (after the arrow).

If used with FILE, colno places the cursor relative to the beginning of a file line (after the prefix area). If not specified and the cursor is currently within the file area, it is placed in the same column number in the specified file line. Otherwise, the cursor is placed in the first column.

If used with SCREEN, colno places the cursor relative to the beginning of the logical screen (under the first character of the prefix area). When colno is not specified for CURSOR SCREEN, the default cursor position is the current column in which it is displayed. If the cursor is on the command line, the colno is 7.

Priority *n*
is the priority number assigned. It must be greater than or equal to zero and less than 256. Higher numbers denote higher priorities and lower numbers denote lower priorities. If no priority is specified, all priorities of prefix subcommands and macros are ignored and the last CURSOR subcommand issued positions the cursor.

Initial Setting

'ENTER' key	30 (initial setting for 'SET ENTER')
Change on the screen	20

Refer to Chapter 4: "Prefix Subcommands and Macros" in this publication for a complete list of the priorities for all prefix subcommands and macros.

Usage Notes

1. The CURSOR subcommand with the FILE or SCREEN operands is mainly intended to be issued from XEDIT macros.
2. You can display the current cursor position, relative to the beginning of the file and the beginning of the screen, with the QUERY CURSOR subcommand (or, within a macro, you can use the EXTRACT subcommand).
3. CURSOR HOME (and CURSOR COLUMN) are good candidates for PF key assignment. The initial setting of the PF12 key is CURSOR HOME.

   If CURSOR HOME is issued and the cursor cannot be returned to its previous position in the file (for example, if scrolling or splitting the screen removes that part of the file from the current screen), the cursor is positioned in the current column of the current line. However, if the cursor was located in an area other than the file area (like a reserved line) and CURSOR HOME is issued, the cursor returns to its position on the screen even if the screen is scrolled (provided a change to the screen layout does not make this impossible, for example, by splitting the screen).

   If the cursor is not on any logical screen, it moves to the command line. HOME, at this point, is line one, column one.
4. If you issue CURSOR HOME immediately after entering XEDIT, the cursor will move to the current line and column if they are on the logical screen. If the current line and column are not on the screen, the cursor will move to the first verify column.
5. If the cursor is in the prefix area when CURSOR HOME is issued, the cursor will move to the command line. If you issue CURSOR HOME again, the cursor will move to the first verify column within the line adjacent to that prefix area.
6. If the cursor is on a valid spot on the virtual screen, but that spot is not displayed on the physical screen, the cursor will be repositioned according to the rules in the *VM/SP CMS Command Reference* for the CURSOR VSCREEN command.

Notes For Macro Writers

1. The CURSOR subcommand does not scroll the screen. Therefore, when you use CURSOR FILE lineno, the line number (lineno) must appear on the current logical screen. For example, in the following subcommand,

   ```
   cursor file 700
   ```

   file line 700 must appear on the current logical screen.

   When you use CURSOR SCREEN lineno, the screen line (lineno) must be within your logical screen size.

2. A macro can indicate where the cursor should be positioned as well as the priority for this cursor movement. As a result, when multiple prefix subcommands are issued and a macro is executed on the command line, the cursor is positioned at the location specified with the highest priority.

   Note: The cursor position is ONLY updated when a higher cursor priority is encountered. Therefore, if two prefix subcommands are specified that have the same cursor priority, the first one will determine the cursor placement. Likewise, if a prefix subcommand is issued and a macro is executed on the command line, both with equal cursor priorities, the prefix subcommand will determine the cursor placement.

3. If multiple CURSOR subcommands are issued, the priority of each is checked and the cursor position is updated to reflect the new setting, which corresponds to the command with the highest priority. The cursor remains in the screen it was in when the ENTER key is pressed; therefore, a CURSOR subcommand in another logical screen will have no effect on cursor placement.

Messages

520E	Invalid operand: *operand* [RC=5]
521E	Invalid line number [RC=1 or 5]
527E	Invalid column number [RC=1]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
1	Specified line (lineno) or column (colno) will set the cursor outside of the screen — no action taken
3	Subcommand valid only for display terminal
5	Invalid or missing operand or (line) number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

DELETE

Use the DELETE subcommand to delete one or more lines from a file, beginning with the current line.

Format

DELete	[*target* <u>1</u>]

Operand

target
defines the number of lines to be deleted. Deletion starts with the current line and continues up to, but does not include, the target line. If you enter an asterisk (*), the rest of the file is deleted. If you omit target, only the current line is deleted.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Note

1. You can clear a line by pressing the spacebar once in column one and then pressing the ERASE EOF key (or repeatedly pressing the DELETE key). If a line is cleared in this manner, a blank line remains in the file. If you don't press the spacebar, the data will come back in the line the next time you press the ENTER key. This prevents you from erasing an entire line accidentally.

Responses

If the operand is specified as any type of target other than a relative displacement target or if the Top of File or End of File line is reached, the number of lines deleted is displayed with the following message:

```
501I nn line(s) deleted
```

On a forward DELETE (toward the end of the file) the line immediately following the last line deleted becomes the new current line.

On a backward DELETE (toward the top of the file), the line preceding the last deleted line becomes the new current line.

If you delete all lines in a file, the following message is displayed:

```
559W Warning: file is empty
```

Messages

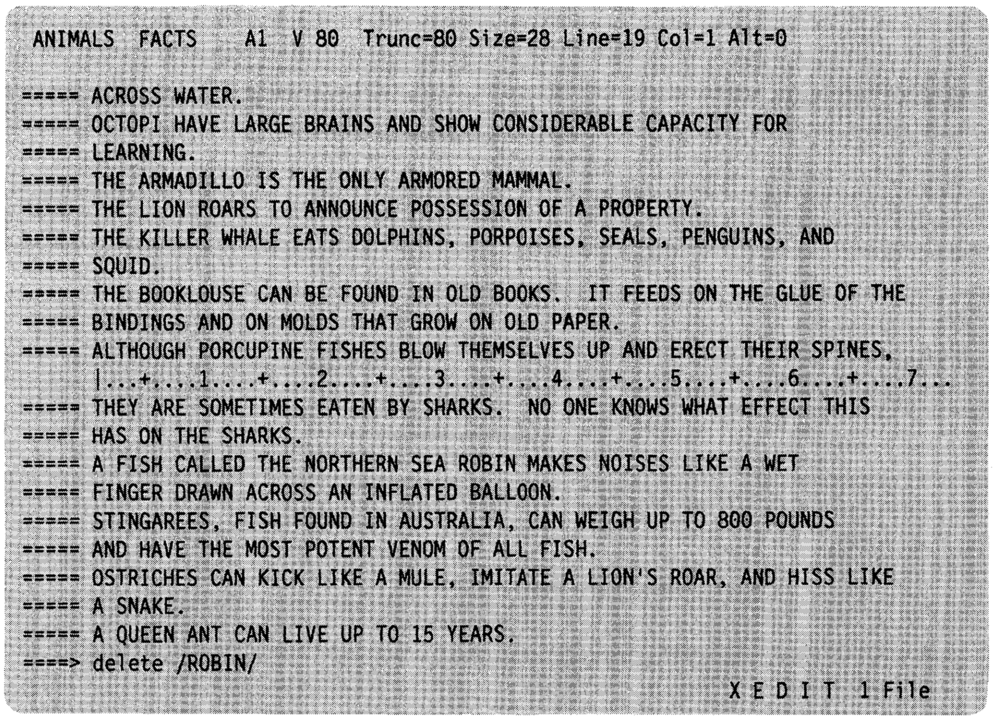
520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]

Return Codes

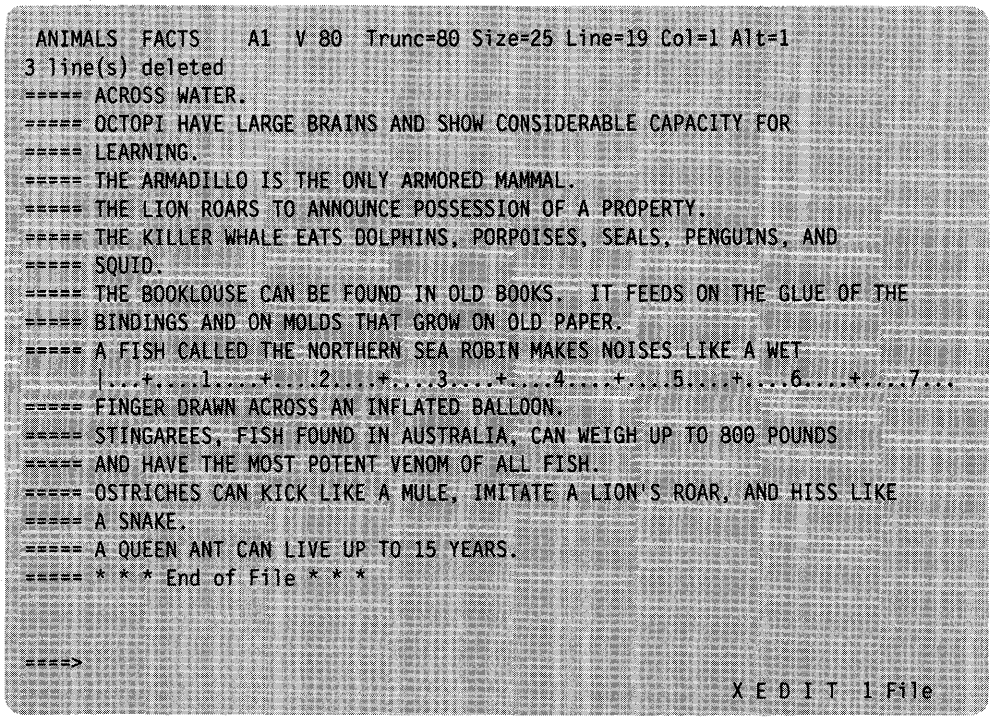
0	Normal
1	Partial delete due to TOF or EOF reached during execution
2	Target line not found
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 7 is a before-and-after example of the DELETE subcommand.



```
 ANIMALS  FACTS    A1  V 80  Trunc=80 Size=28 Line=19 Col=1 Alt=0

===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
===== THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS.  IT FEEDS ON THE GLUE OF THE
===== BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
===== HAS ON THE SHARKS.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
===== FINGER DRAWN ACROSS AN INFLATED BALLOON.
===== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
===== AND HAVE THE MOST POTENT VENOM OF ALL FISH.
===== OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISS LIKE
===== A SNAKE.
===== A QUEEN ANT CAN LIVE UP TO 15 YEARS.
====> delete /ROBIN/
                                                             X E D I T  1 File
```



```
 ANIMALS  FACTS    A1  V 80  Trunc=80 Size=25 Line=19 Col=1 Alt=1
3 line(s) deleted
===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
===== THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS.  IT FEEDS ON THE GLUE OF THE
===== BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== FINGER DRAWN ACROSS AN INFLATED BALLOON.
===== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
===== AND HAVE THE MOST POTENT VENOM OF ALL FISH.
===== OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISS LIKE
===== A SNAKE.
===== A QUEEN ANT CAN LIVE UP TO 15 YEARS.
===== * * * End of File * * *


====>
                                                             X E D I T  1 File
```

Figure 7. The DELETE Subcommand — Before and After

DOWN

Use the DOWN subcommand to advance the line pointer a specified number of lines toward the end of the file. The line pointed to becomes the new current line. (The DOWN subcommand is equivalent to the NEXT subcommand.)

Format

Down	[*n* | * $\underline{1}$]

Operand

n

is the number of lines to move the line pointer. If you specify an asterisk (*), the line pointer moves to the "End of File" line. If you omit n, the pointer moves down only one line.

Usage Note

The Down n subcommand is equivalent to a plus (+) target definition.

For example:

```
down 3
```

is equivalent to

```
+3
```

Response

The line pointed to becomes the new current line.

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]

Return Codes

0	Normal
1	End of file reached and displayed
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

Refer to the example of the NEXT subcommand.

DUPLICAT

Use the DUPLICAT subcommand to duplicate one or more lines, starting with the current line, for a specified number of times. The new line(s) is inserted immediately after the original line(s).

Format

DUPlicat	$\left[\begin{array}{c} n \\ \underline{1} \end{array} \left[\begin{array}{c} target \\ \underline{1} \end{array}\right]\right]$

Operands

n
specifies the number of times that the line(s) is to be duplicated. If you omit n, the current line is duplicated once.

target
defines the number of lines to be duplicated. Duplication starts with the current line and continues up to but does not include the target line. If you omit target, only the current line is duplicated.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Response

The last line duplicated becomes the new current line.

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
546E	Target not found [RC=2]
557S	No more storage to insert lines [RC=4]

Return Codes

0	Normal
1	TOF or EOF reached
2	Target line not found
4	No more storage to continue duplicating
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

EMSG

The EMSG subcommand has two formats. Use the first format to display a message at the terminal and to sound the alarm. Use the second format in macros and modules that interface with XEDIT and whose messages follow the rules for VM/SP messages (see the *VM/SP System Messages and Codes* book). The severity of the message determines whether or not the alarm is sounded.

Format

EMSG	[*text*]
	[*mmmnnn*[*n*] s]

Operands

text
is the text of the message to be displayed. If no text is specified, a blank line will be displayed.

mmmnnn[n]s
is the message identification.

mmm
are three letters indicating which macro or module generated the message.

nnn[n]
is the three-digit or four-digit message number. It may be used to find additional information in the *VM/SP System Messages and Codes* book.

s
indicates the severity and is one of the following:

R - response	E - error
I - information	S - severe error
W - warning	T - terminal (unrecoverable) error

With a severity of R, I, or W, the alarm is not sounded. With a severity of E, S, or T, the alarm is sounded when the message is displayed.

The message is prefixed by "DMS" before being displayed.

Usage Notes

1. The SOS ALARM subcommand may be used to sound the alarm without displaying any message.
2. Messages are displayed according to the SET MSGMODE setting (ON or OFF).
3. When using the second format, the message identification is processed according to the CP EMSG setting (see the *VM/SP CP General User Command Reference*). In addition, the message is processed according to the SET MSGMODE setting (LONG or SHORT).
4. The message identification (*mmmnnns*) must not contain imbedded blanks. It must be preceded by only one blank delimiter.

5. EMSG can also be used to display messages on a typewriter terminal.

Response

The message is displayed in the message area of the logical screen.

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

EXPAND

Use the EXPAND subcommand to reposition data in one or more file lines that contain tab characters (X'05'), according to the current tab settings (defined by a SET TABS subcommand). Each time a tab character appears in a line, the editor repositions the data in the column defined by the SET TABS subcommand.

Format

EXPand	[*target* |1]

Operand

target
defines the number of lines to be expanded. Lines are expanded starting with the current line and continuing up to, but not including, the target line. If you enter an asterisk (*), the rest of the file is expanded. If you omit target, only the current line is expanded.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Notes

1. In order for EXPAND to work properly, lines containing backspace characters (X'16') must have been entered into the file with SET IMAGE CANON in effect, that is, compound characters ordered with backspaces arranged singly between the characters that overlay each other.
2. Refer to the COMPRESS subcommand description for an explanation of how to use the COMPRESS, EXPAND, and SET TABS subcommands to reposition data.
3. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.

Responses

If you specify that an EXPAND is to occur on multiple lines and it does occur, the current line pointer will be:

a. Unchanged if SET STAY ON has been issued
b. Moved to the last line expanded, if SET STAY OFF is in effect (the default).

Messages

504E	*nn* line(s) {truncated|spilled} [RC = 3]
520E	Invalid operand: *operand* [RC = 5]
546E	Target not found [RC = 2]
581E	Subcommand is not valid in extended mode [RC = 3]
585E	No line(s) changed [RC = 1 or 4]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
3	Truncated or spilled, or subcommand is not valid in extended mode
4	No line(s) changed
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

See the "Examples" section of the COMPRESS subcommand.

EXTRACT

Use the EXTRACT subcommand *within a macro* to get information about internal XEDIT variables or about file data. XEDIT returns the information in one or more variables, which can then be used or examined by the macro. Operands of the EXTRACT subcommand are separated by a self-defining delimiter in the same fashion that string targets or operands are delimited.

Format

EXTract	*/operand* [*/operand* [*/operand* . . .]]

Operands

/(diagonal)
signifies any delimiting character that does not appear in the "operand" string(s).

operand
may be any one of the keywords listed below.

```
ACTion          FLscreen    NULls           TABLine
ALT             FMode       NUMber          TABS
APL             FName       PA [n|*]        TARGet
ARBchar         FType       PACK            TERMinal
AUtosave        FULLread    PENDing (below) TEXT
BASEft          HEX         PF [n|*]        TOF
BRKkey          IMage       Point [*]       TOFEOF
CASE            IMPcmscp    PREfix (below)  TOL
CMDline         INPmode     RANge           TRANSLat
COLOR field|*   LASTLorc    RECFm           TRunc
COLPtr          LASTmsg     REMOte          UNIQueid
COLumn          LENgth      RESERved [*]    UNTil
CTLchar [char]  LIBName     RING            UPDate
CURLine         LIBType     SCALe           VARblank
CURSor          LIne        SCOPE           Verify
DISPlay         LINENd      SCReen          VERShift
EDIRName        LOCk        SELect          Width
EFMode          LRecl       Seq8            WINdow
EFName          LScreen     SERial          WRap
EFType          MACRO       SHADow          Zone
ENTer           MASK        SIDcode         =
EOF             MEMber      SIZe
EOL             MSGLine     SPAN
ESCape          MSGMode     SPILL
ETARBCH         NBFile      STAY
ETMODE          NBScope     STReam
FILler          NONDisp     SYNonym [name|*]

PENDing [BLOCK][OLDNAME] name|* [target1[target2]]
PREfix [Synonym n|Synonym *]
```

The following is a list of the values returned by EXTRACT for each of the settings. The variables are returned in the form "name.n" where "name" is the full name of the variable requested and "n" is a subscript that distinguishes the different values returned. An exception occurs in the case of =. Variables are returned as "EQUALSIGN.", followed by the number. In all cases the value of "name.0" is the number of variables returned for that setting. EXEC 2 variables that are returned are preceded by an ampersand (&).

ACTion

returns "ON" or "OFF" to indicate whether any action other than displaying or scrolling has been taken on this file. This includes any file ID change, or file characteristic change (LRECL, RECFM, PACK, SERIAL, SIDCODE, or ALT), as well as any other changes made to the file.

```
ACTION.0       number of variables returned
      .1       ON|OFF
```

ALT

returns the number of alterations that have been made to the file since the last AUTOSAVE and SAVE, or as specified in the SET ALT subcommand.

```
   ALT.0       number of variables returned
      .1       number of changes since last AUTOSAVE
      .2       number of changes since last SAVE
```

APL

returns "ON" or "OFF" as defined by the SET APL subcommand.

```
   APL.0       number of variables returned
      .1       ON|OFF
```

ARBchar

returns "ON" or "OFF" and the arbitrary character specified in the SET ARBCHAR subcommand.

```
ARBCHAR.0      number of variables returned
       .1      ON|OFF
       .2      arbitrary character
```

AUtosave

returns the current setting defined in the SET AUTOSAVE subcommand: the AUTOSAVE count, file ID, the number of alterations, and the disk or directory on which the autosave file is written.

```
AUTOSAVE.0     number of variables returned
        .1     OFF or autosave count
        .2     autosave file name
        .3     number of alterations since last
                autosave
        .4     autosave file mode (one character)
```

BASEft

returns the base file type specified in the LOAD subcommand or the XEDIT command (where the LOAD is implicit).

```
BASEFT.0       number of variables returned
      .1       file type specified in the LOAD
                subcommand or the XEDIT command
                (where the LOAD is implicit)
                when XEDIT is invoked
```

BRKkey

returns "ON" or "OFF" as defined by the CP terminal brkkey setting. If ON, then also returns the key.

```
  BRKKEY.0      number of variables returned
        .1      ON|OFF
        .2      PAn|PFn
```

CASE

returns the case setting as defined in the SET CASE subcommand.

```
    CASE.0      number of variables returned
        .1      MIXED|UPPER
        .2      RESPECT|IGNORE
```

CMDline

returns "ON", "OFF", "TOP", or "BOTTOM" as specified in the SET CMDLINE subcommand and the line number designated as the command line on the screen. CMDLINE.2 is not returned when CMDLINE.1 = OFF.

```
 CMDLINE.0      number of variables returned
        .1      ON|OFF|TOP|BOTTOM
        .2      line number on screen
```

COLOR field|*
COLOR field

returns area of screen and its respective color, extended highlighting, highlighting, and programmed symbol set options as specified in the SET COLOR subcommand. The field may be specified as: Arrow, Cmdline, CUrline, Filearea, Idline, Msgline, Pending, PRefix, Scale, SHadow, STatarea, Tabline, TOfeof.

```
   COLOR.0      number of variables returned
        .1      color
        .2      extended highlighting
        .3      HIGH|NOHIGH
        .4      programmed symbol set
```

COLOR *

returns the areas of the screen and their respective colors, extended highlighting, highlighting, and programmed symbol set options as they have been specified in the SET COLOR subcommand.

```
   COLOR.0      number of variables returned
        .1      ARROW     color exthi HIGH|NOHIGH PSs
        .2      CMDLINE   color exthi HIGH|NOHIGH PSs
        .3      CURLINE   color exthi HIGH|NOHIGH PSs
        .4      FILEAREA  color exthi HIGH|NOHIGH PSs
        .5      IDLINE    color exthi HIGH|NOHIGH PSs
        .6      MSGLINE   color exthi HIGH|NOHIGH PSs
        .7      PENDING   color exthi HIGH|NOHIGH PSs
        .8      PREFIX    color exthi HIGH|NOHIGH PSs
        .9      SCALE     color exthi HIGH|NOHIGH PSs
        .10     SHADOW    color exthi HIGH|NOHIGH PSs
        .11     STATAREA  color exthi HIGH|NOHIGH PSs
        .12     TABLINE   color exthi HIGH|NOHIGH PSs
        .13     TOFEOF    color exthi HIGH|NOHIGH PSs
```

COLPtr

returns "ON" or "OFF" as defined by the SET COLPTR subcommand.

```
  COLPTR.0      number of variables returned
        .1      ON|OFF
```

COLumn

returns the column number of the column pointer.

COLUMN.0	number of variables returned
.1	current column number

CTLchar [char]
CTLchar char

returns whether or not "char" is in a protected status and the color, extended highlighting, highlighting, and programmed symbol set associated with "char" as specified by the SET CTLCHAR subcommand. When "char" is specified and CTLCHAR.1 = OFF, then CTLCHAR.2 through CTLCHAR.5 are not returned. If SET CTLCHAR is not OFF (there are control characters defined) and just this particular character is not defined, then CTLCHAR.0 will be 0 and no other variables will be returned.

CTLCHAR.0	number of variables returned
.1	PROTECT|NOPROTECT|OFF (OFF means no "chars" defined)
.2	color
.3	extended highlighting
.4	HIGH|NOHIGH|INVISIBLE
.5	programmed symbol set

CTLchar

returns the escape character and all control characters, if any, defined by the SET CTLCHAR subcommand. If CTLCHAR.1 = "OFF", then CTLCHAR.2 and CTLCHAR.3 are not returned.

CTLCHAR.0	number of variables returned
.1	ON|OFF
.2	escape character
.3	list of control characters (if any)

CURLine

If you are using a display terminal, EXTRACT /CURLINE/ returns the line number of the current line as specified by the SET CURLINE subcommand, the contents of the current line from column one through the truncation column (excluding trailing blanks), and whether or not it has been changed or inserted during this editing session.

CURLINE.0	number of variables returned
.1	M [+n|-n] | [-] n (M = middle of screen)
.2	actual line number on screen
.3	contents of current line (or null if line pointer at TOF or EOF line)
.4	ON|OFF (ON if CURLINE has been changed or inserted in this editing session)
.5	OLD|OLD CHANGED|NEW|NEW CHANGED (OLD if CURLINE not inserted this session, NEW if CURLINE inserted this session, and CHANGED if CURLINE changed during this session)

If you are using a typewriter terminal, EXTRACT /CURLINE/ returns the contents of the current line from column one through the truncation column (excluding trailing blanks), and whether or not it has been changed or inserted during this editing session.

```
CURLINE.0      number of variables returned
       .1      -1
       .2      -1
       .3      contents of current line
                (or null if line pointer at
                TOF or EOF line)
       .4      ON|OFF (ON if CURLINE has been changed
                or inserted in this editing session)
       .5      OLD|OLD CHANGED|NEW|NEW CHANGED
                (OLD if CURLINE not inserted this
                 session, NEW if CURLINE inserted
                 this session, and CHANGED if
                 CURLINE changed during this
                 session)
```

CURSor

returns the current and the original position of the cursor on the screen (line number and column number), the current and the original position of the cursor in the file (line number and column number), and the priority number (if assigned) as specified in the CURSOR subcommand. The current position is the position where the cursor would be placed if the screen were displayed at this time. The current position reflects relative changes due to additions/deletions of lines resulting from prefix execution. It does not necessarily reflect where the cursor will eventually be located when the screen is actually displayed. This is because cursor priority cannot be resolved until the screen is displayed. The original position is the position of the cursor when the screen was read, which has been updated with changes due to the execution of prefix subcommands and macros. If the cursor is not in the file, then CURSOR.3 and CURSOR.4 = -1. Likewise, if the cursor was not originally in the file, then CURSOR.7 and CURSOR.8 = -1. However, if the screen had not been displayed, then CURSOR.7 and CURSOR.8 = 0. This can occur if EXTRACT/CURSOR/ is issued from the profile macro.

```
CURSOR.0    number of variables returned
      .1    position of cursor on screen (line number)
      .2    position of cursor on screen (col. number)
      .3    position of cursor in file   (line number)
      .4    position of cursor in file   (col. number)
      .5    original .1--original position of the
                         cursor on the screen
                         (line number)
      .6    original .2--original position of the
                         cursor on the screen
                         (column number)
      .7    original .3--original position of the
                         cursor in the file
                         (line number)
      .8    original .4--original position of the
                         cursor in the file
                         (column number)
      .9    highest priority or zero
```

DISPlay

returns the range of selection levels which are included in the display, as specified by the SET DISPLAY subcommand.

```
DISPLAY.0      number of variables returned
       .1      start of display range
       .2      end of display range
```

EDIRName

returns the name of the SFS directory containing the file at the time the file was first loaded.

```
EDIRNAME.0    number of variables returned
        .1    entry directory name (directory name when the
               XEDIT environment is entered), or null for a
               minidisk file.
```

EFMode

returns the two-character file mode of the file at the time the file was first loaded.

```
EFMODE.0    number of variables returned
      .1    entry file mode (file mode when the XEDIT
             environment is entered)
```

EFName

returns the eight-character file name of the file at the time the file was first loaded.

```
EFNAME.0    number of variables returned
      .1    entry file name (file name when the XEDIT
             environment is entered)
```

EFType

returns the eight-character file type of the file at the time the file was first loaded.

```
EFTYPE.0    number of variables returned
      .1    entry file type (file type when the XEDIT
             environment is entered)
```

ENTer

returns "BEFORE", "AFTER", "ONLY", or "IGNORE" and the ENTER key definition as specified by the SET ENTER subcommand. If the ENTER key is undefined, then ENTER.0=0 and no other variables are returned.

```
ENTER.0       number of variables returned
     .1       BEFORE|AFTER|ONLY|IGNORE
     .2       ENTER key definition
```

EOF

returns "ON" or "OFF" as determined by the editor. EOF is "ON" when the line pointer reaches the End of File (or End of Range) line.

```
EOF.0       number of variables returned
   .1       ON|OFF
```

EOL

returns "ON" or "OFF" as determined by the editor. EOL is "ON" when the column pointer reaches zone2 + 1.

```
EOL.0       number of variables returned
   .1       ON|OFF
```

ESCape

returns "ON" or "OFF" and the escape character defined by the SET ESCAPE subcommand.

```
ESCAPE.0       number of variables returned
      .1       ON|OFF
      .2       escape character
```

ETARBCH
returns "ON" or "OFF" and the extended arbitrary character specified in the SET ETARBCH subcommand.

```
ETARBCH.0   number of variables returned
       .1   ON|OFF
       .2   extended arbitrary character enclosed
             by a shift-out and a shift-in character
```

ETMODE
returns "ON" or "OFF" as specified by the SET ETMODE subcommand.

```
ETMODE.0       number of variables returned
      .1       ON|OFF
```

FILler
returns the filler character defined by the SET FILLER subcommand.

```
FILLER.0       number of variables returned
      .1       filler character
```

FLscreen
returns the line numbers of the first and last lines of the file displayed on the screen.

```
FLSCREEN.0   number of variables returned
        .1   line number of the first line of the file
              displayed on the screen
        .2   line number of the last line of the file
              displayed on the screen
```

FMode
returns the two-character file mode.

```
FMODE.0       number of variables returned
     .1       file mode
```

FName
returns the eight-character file name.

```
FNAME.0       number of variables returned
     .1       file name
```

FType
returns the eight-character file type.

```
FTYPE.0       number of variables returned
     .1       file type
```

FULLread
returns "ON" or "OFF" as specified by the SET FULLREAD subcommand.

```
FULLREAD.0       number of variables returned
        .1       ON|OFF
```

HEX
returns "ON" or "OFF" as specified in the SET HEX subcommand.

```
HEX.0       number of variables returned
   .1       ON|OFF
```

IMage
returns "ON", "OFF", or "CANON" as specified in the SET IMAGE subcommand.

```
IMAGE.0       number of variables returned
     .1       ON|OFF|CANON
```

IMPcmscp
returns "ON" or "OFF" as specified in the SET IMPCMSCP subcommand.

```
IMPCMSCP.0       number of variables returned
        .1       ON|OFF
```

INPmode
returns "ON" or "OFF" as determined by whether or not you are in input mode.

```
 INPMODE.0       number of variables returned
        .1       ON|OFF
```

LASTLorc
returns the current contents of the last locate or change buffer, as specified either by SET LASTLORC or by the editor.

```
LASTLORC.0       number of variables returned
        .1       contents of LASTLORC buffer or null
```

LASTmsg
returns the last message that was issued by the editor. (Depending on the SET MSGMODE operands specified, this message may or may not have been displayed.) If CP SET EMSG OFF is in effect, LASTMSG is not updated, even if SET MSGMODE ON.

```
 LASTMSG.0       number of variables returned
        .1       last message issued or null
```

LENgth
returns the length of the current line from column one through the truncation column (excluding trailing blanks).

```
  LENGTH.0       number of variables returned
        .1       length of current line (Length
                  of TOF and EOF lines is zero.)
```

LIBName
returns the library file name while editing a member of a CMS library.

```
 LIBNAME.0       number of variables returned
        .1       library file name or blanks (if not
                  editing a member of a library)
```

LIBType
returns the library file type while editing a member of a CMS library.

```
 LIBTYPE.0       number of variables returned
        .1       library file type or blanks (if not
                  editing a member of a library)
```

LIne
returns the current line number, relative to the beginning of the file

```
    LINE.0       number of variables returned
        .1       line number (in the file) of current
                  line
```

LINENd
returns "ON" or "OFF" and the line end character as defined by the SET LINEND subcommand.

```
  LINEND.0       number of variables returned
        .1       ON|OFF
        .2       linend character
```

LOCk

returns "ON" or "OFF" to indicate whether the file was locked by the editor at the time it was first loaded. "ON" will be returned even if the lock has since been dropped by the file system.

```
    LOCK.0       number of variables returned
      .1         ON|OFF
```

LRecl

returns the logical record length of the file.

```
   LRECL.0       number of variables returned
      .1         lrecl of file
```

LScreen

returns six integers--the number of lines and the number of columns in the logical screen, the line number and the column number defining the top left corner of the logical screen on the virtual screen, and the number of lines and the number of columns of the virtual screen.

```
LSCREEN.0   number of variables returned
       .1   number of lines on logical screen
       .2   number of columns on logical screen
       .3   line number of top left of logical screen
       .4   column number of top left of logical
             screen
       .5   number of lines in virtual screen
       .6   number of columns in virtual screen
```

MACRO

returns "ON" or "OFF" as specified by the SET MACRO subcommand.

```
   MACRO.0       number of variables returned
      .1         ON|OFF
```

MASK

returns the current mask line as defined by the SET MASK subcommand.

```
    MASK.0       number of variables returned
      .1         mask definition
```

MEMber

returns "ON" if you are editing a member of a CMS library (the MEMBER option was specified when you entered XEDIT) or "OFF" if you are not editing a member of a CMS library.

```
  MEMBER.0       number of variables returned
      .1         ON|OFF
```

MSGLine

returns "ON" or "OFF", the location of the message line, the number of lines the message line can expand to, and OVERLAY, if specified, as defined by the SET MSGLINE subcommand. If MSGLINE.1 = OFF, then MSGLINE.2 through MSGLINE.4 are not returned.

```
 MSGLINE.0       number of variables returned
      .1         ON|OFF
      .2         M [+n|-n] | [-] n
                  (M = middle of screen)
      .3         number of lines the message line can
                  expand to for displaying a message
      .4         OVERLAY or nulls
```

MSGMode

returns "ON" or "OFF", and "LONG" or "SHORT" as defined by the SET MSGMODE subcommand.

```
MSGMODE.0      number of variables returned
       .1      ON|OFF
       .2      LONG|SHORT
```

NBFile

returns the number of files you are currently editing.

```
 NBFILE.0      number of variables returned
       .1      number of files in XEDIT ring
```

NBScope

returns the number of lines in the file within the current scope setting and the position of the current line within that number. If SCOPE is set to ALL, then the size of the file and the number of the current line is returned. If SCOPE is DISPLAY, then the number of lines that fall within the defined DISPLAY level and the count of the current line within this set of lines is returned. If the Top of File line is the current line, NBSCOPE.2=0. If the End of File line is the current line, NBSCOPE.2=NBSCOPE.1+1.

```
NBSCOPE.0     number of variables returned
       .1     number of lines within the current
               scope
       .2     position of the current line within
               the scope
```

NONDisp

returns the character defined by the SET NONDISP subcommand.

```
NONDISP.0      number of variables returned
       .1      nondisp character
```

NULls

returns "ON" or "OFF" as specified by the SET NULLS subcommand.

```
  NULLS.0      number of variables returned
       .1      ON|OFF
```

NUMber

returns "ON" or "OFF" as specified by the SET NUMBER subcommand.

```
 NUMBER.0      number of variables returned
       .1      ON|OFF
```

PA [n|*]
PAn

returns "BEFORE", "AFTER", "ONLY", or "IGNORE" and the PAn key definition as specified by the SET PAn subcommand. If the PAn key is undefined, then PAn.0=0 and no other variables are returned.

```
    PAn.0      number of variables returned
    PAn.1      BEFORE|AFTER|ONLY|IGNORE
    PAn.2      PAn key definition
```

PA [*]

returns "BEFORE", "AFTER", "ONLY", or "IGNORE" and the PAn key definition as specified in the SET PAn subcommands for all of the PA keys. If any PA key is undefined, then PAn.0=0 and no other variables are returned for that PA key.

PA1.0	number of variables returned for PA1
PA1.1	BEFORE|AFTER|ONLY|IGNORE
PA1.2	PA1 key definition
PA2.0	number of variables returned for PA2
PA2.1	BEFORE|AFTER|ONLY|IGNORE
PA2.2	PA2 key definition
PA3.0	number of variables returned for PA3
PA3.1	BEFORE|AFTER|ONLY|IGNORE
PA3.2	PA3 key definition

PACK

returns "ON" or "OFF" as specified by the SET PACK subcommand.

PACK.0	number of variables returned
.1	ON|OFF

PENDing [BLOCK] [OLDNAME] *name*|* **[***target1***[***target2***]]**
[:1 [*]]

returns information from the "pending list" (see SET PENDING in this publication) with respect to a particular subcommand or macro name or the first pending entry.

BLOCK

indicates that the pending list is to be checked for BLOCK entries only and will return the word "BLOCK" if one is found. "BLOCK" will also be returned if a block entry is the one found regardless of whether BLOCK was specified.

OLDNAME

indicates that the name specified is the original name of the prefix subcommand or macro.

name

indicates the name of the prefix subcommand or macro for which you are searching. If OLDNAME is also specified, name must be the original name of the prefix subcommand or macro, regardless of whether or not a synonym has been assigned to that name. Otherwise, it is assumed to be a synonym (that is, a new name) or a name without a synonym.

*

indicates to return the first entry in the pending list. If BLOCK is also specified, * indicates to return the first block entry.

target1 target2

indicates the range in the file where the associated prefix subcommand or macro must be located. If only target1 is specified, the search will be conducted starting at target1 and will run to the end of the file. Target1 will be obtained relative to the top of the file. For example, EXTRACT /PENDING * +3 / will start at the top of the file and go 3 lines forward in the file to begin the search. Target2 will be obtained relative to target1. If target1 is line 3 and you enter a +5 as target2, then the search will be from line 3 through line 8, inclusive of both target lines. If target2 is specified prior to target1, the result will be the same as if you had specified target1 target2. For example, EXTRACT

/PENDING * :10 :2/ will produce the same results as EXTRACT /PENDING * :2 :10/. If no targets are specified, the entire pending list is searched, that is, the entire file is searched, starting from the top of the file (:1).

If no pending entry is found, then PENDING.0=0 and no other variables are returned. If a target is specified and is not found, the return code from EXTRACT will be "2".

```
PENDING.0     number of variables returned
      .1      line number in the file
      .2      newname--the name entered in the prefix
                       area
      .3      oldname--the original name of the prefix
                       subcommand or macro, after
                       synonym resolution
      .4      BLOCK|null--BLOCK is returned if a prefix
                          block entry is located in the
                          pending list; otherwise,
                          nulls are returned.
      .5      op1|null--the first operand accompanying
                        the subject prefix subcommand
                        or macro
      .6      op2|null--the second operand accompanying
                        the subject prefix subcommand
                        or macro
      .7      op3|null--the third operand accompanying
                        the subject prefix subcommand
                        or macro
```

PF [n|*]
PFn

returns "BEFORE", "AFTER", "ONLY", or "IGNORE" and the PFn key definition as specified in the SET PFn subcommand. If the PFn key is undefined, then PFn.0=0 and no other variables are returned.

```
PFn.0      number of variables returned
PFn.1      BEFORE|AFTER|ONLY|IGNORE
PFn.2      PFn key definition
```

PF [*]

returns "BEFORE", "AFTER", "ONLY", or "IGNORE" and the PFn key definition as specified by the SET PFn subcommands for all of the PF keys. If any PF key is undefined, then PFn.0=0 and no other variables are returned for that PF key.

```
 PF1.0      number of variables returned
             for PF1
 PF1.1      BEFORE|AFTER|ONLY|IGNORE
 PF1.2      PF1 key definition
 PF2.0      number of variables returned
             for PF2
 PF2.1      BEFORE|AFTER|ONLY|IGNORE
 PF2.2      PF2 key definition
   .          .
   .          .
   .          .
PF24.0      number of variables returned
             for PF24
PF24.1      BEFORE|AFTER|ONLY|IGNORE
PF24.2      PF24 key definition
```

Point [*]

Point

returns the symbolic name(s) associated with the current line. If no names have been assigned to the current line, then POINT.0=0 and POINT.1 is not returned.

```
POINT.0       number of variables returned
     .1       line number and up to the last 100
               names assigned to the current line
```

Point *

returns all symbolic names that have been defined, starting at the top of the file. If no names are defined, then POINT.0=0 and no other variables are returned.

```
POINT.0       number of variables returned
     .1       line number and all names on
               first named line
     .2       line number and all names on
               second named line
      .       .
      .       .
      .       .
POINT.n       line number and all names on
               nth named line
```

PREfix [Synonym *name*|Synonym *]

PREfix

returns "ON", "OFF", or "NULLS" and "RIGHT" or "LEFT" as specified in the SET PREFIX subcommand.

```
PREFIX.0      number of variables returned
      .1      ON|OFF|NULLS
      .2      RIGHT|LEFT
```

PREfix Synonym *name*

returns the original name associated with the prefix subcommand or macro, before synonym resolution. If "name" is not in the prefix synonym table, then oldname=name.

```
PREFIX.0      number of variables returned
      .1      oldname
```

PREfix Synonym *

returns both the old and the new names of the synonyms defined for the prefix subcommand(s) or macro(s).

```
PREFIX.0      number of variables returned
      .1      newname oldname
       .      .
       .      .
PREFIX.n      newname oldname
```

RANge

returns the line numbers of the top and bottom of the range defined by the SET RANGE subcommand.

```
RANGE.0       number of variables returned
     .1       line number of the first line
               in range
     .2       line number of the last line
               in range
```

RECFm
returns the record format, “F”, “V”, “FP”, or “VP”, defined by the SET RECFM subcommand.

```
RECFM.0      number of variables returned
     .1      record format of file
```

REMOte
returns “ON” or “OFF” depending upon whether or not a remote terminal is being used or upon the setting defined by the SET REMOTE subcommand.

```
REMOTE.0      number of variables returned
      .1      ON|OFF
```

RESERved [*]
RESERved
returns a list of line numbers of screen lines currently reserved. If no RESERVED lines have been defined, then RESERVED.0=0 and no other variables are returned.

```
RESERVED.0    number of variables returned
        .1    list of reserved line numbers
```

RESERved *
returns the line numbers of the screen lines currently reserved and the colors, extended highlighting, programmed symbol set, highlighting, and text associated with those reserved lines as specified by the SET RESERVED subcommand. If no RESERVED lines have been defined, then RESERVED.0=0 and no other variables are returned.

```
RESERVED.0      number of variables returned
        .1      linenum color exthi PSs HIGH|NOHIGH text
        .2      linenum color exthi PSs HIGH|NOHIGH text
         .
         .
         .
        .n      linenum color exthi PSs HIGH|NOHIGH text
```

RING
returns the number of files you are editing and the file identification line for each file.

```
RING.0      number of variables returned
    .1      number of files in the ring
    .2      file identification line of the
             first file
    .3      file identification line of the
             second file
     .      .
     .      .
     .      .
    .n      file identification line of the
             nth-1 file
```

SCALe

returns "ON" or "OFF" and the position of the SCALE as specified by the SET SCALE subcommand (or SCALE prefix subcommand) and the line number of the scale on the screen. If SCALE is "OFF," only SCALE.0, SCALE.1, and SCALE.2 are returned.

```
SCALE.0     number of variables returned
     .1     ON|OFF
     .2     M [+n|-n] | [-] n
             (M = middle of screen)
     .3     line number on screen
```

SCOPE

returns "DISPLAY" or "ALL" as specified by the SET SCOPE subcommand.

```
SCOPE.0     number of variables returned
     .1     ALL|DISPLAY
```

SCReen

returns the attributes of the screens that have been defined by the SET SCREEN subcommand.

```
SCREEN.0     number of variables returned
      .1     SIZE|WIDTH|DEFINE screen definition
```

SELect

returns the selection level of the current line and the maximum selection level for the file as specified by the SET SELECT subcommand.

```
SELECT.0     number of variables returned
      .1     selection level of current line
      .2     maximum selection level in the file
```

Seq8

returns "OFF" if the XEDIT command or LOAD subcommand was issued with the NOSEQ8 operand; if not, returns "ON".

```
SEQ8.0     number of variables returned
    .1     ON|OFF
```

SERial

returns the serial identification, the increment value, and the serial number starting value as defined by the SET SERIAL subcommand.

```
SERIAL.0     number of variables returned
      .1     serial or OFF
      .2     increment
      .3     start number
```

SHADow

returns "ON" or "OFF" as specified by the SET SHADOW subcommand.

```
SHADOW.0     number of variables returned
      .1     ON|OFF
```

SIDcode

returns the eight-character string specified in the SIDCODE option of the XEDIT command, the LOAD subcommand, or the SET SIDCODE subcommand.

```
SIDCODE.0     number of variables returned
       .1     eight-character sidcode
               string(if specified) or blanks
```

SIZe
returns the number of records in the file being edited.

```
SIZE.0      number of variables returned
    .1      number of records in file
```

SPAN
returns "ON" or "OFF", "BLANK" or "NOBLANK", and n as defined in the SET SPAN subcommand.

```
SPAN.0      number of variables returned
    .1      ON|OFF
    .2      BLANK|NOBLANK
    .3      n--number of consecutive file lines
               that a character string can span
```

SPILL
returns "ON", "OFF", or "WORD" as defined by the SET SPILL subcommand.

```
SPILL.0     number of variables returned
     .1     ON|OFF|WORD
```

STAY
returns "ON" or "OFF" as specified in the SET STAY subcommand.

```
STAY.0      number of variables returned
    .1      ON|OFF
```

STReam
returns "ON" or "OFF" as specified in the SET STREAM subcommand.

```
STREAM.0    number of variables returned
      .1    ON|OFF
```

SYNonym
returns "ON" or "OFF" as specified in the SET SYNONYM subcommand.

```
SYNONYM.0   number of variables returned
       .1   ON|OFF
```

SYNonym *name*|*
SYNonym *name*
returns the synonym, its minimum abbreviation (returned as the number of letters in the minimum abbreviation), the associated synonym definition, and the linend character, if it has been specified. If no synonym is defined, then SYNONYM.1 and SYNONYM.3 are set equal to the name of the synonym, SYNONYM.2 is set to the length of the name and SYNONYM.4 is set to null.

```
SYNONYM.0    number of variables returned
       .1    name
       .2    length of minimum abbreviation
       .3    definition
       .4    linend character (if specified) or null
```

SYNonym *

returns for each defined synonym its name, its minimum abbreviation (returned as the number of letters in the minimum abbreviation), and the associated synonym definition (that is, everything that was specified in the SET SYNONYM subcommand).

```
SYNONYM.0      number of variables returned
       .1      name abbrev. [LINEND char]
                definition
       .2      name abbrev. [LINEND char]
                definition
        .       .
        .       .
       .n      name abbrev. [LINEND char]
                definition
```

TABLine

returns "ON" or "OFF", and the position of the TABLINE as specified by the SET TABLINE subcommand (or TABL prefix subcommand) and the line number of the tabline on the screen. If TABLINE is "OFF", only TABLINE.0 and TABLINE.1 are returned.

```
TABLINE.0      number of variables returned
       .1      ON|OFF
       .2      M [+n|-n] | [-] n
       .3      line number on screen
```

TABS

returns the tab column numbers defined by the SET TABS subcommand.

```
   TABS.0      number of variables returned
       .1      tab columns
```

TARGet

returns the following information about the character string that matches the last target located via a LOCATE or CLOCATE: line and column number of the first character in the string, and line and column number of the last character in the string.

returns the following information about targets that have been specified as an absolute line number, a relative displacement from the current line, or a line name: line number and current column position (twice).

If the last target located was specified with "&", then only information about the last string found is returned. For example, if the last target located was a result of issuing the following command:

```
LOCATE /a/ & /try/
```

and the line located,

```
This try is even a better one
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

is on line 11 in the file and begins in column 1, EXTRACT /TARGET/ would return the following values.

```
TARGET.0=4
TARGET.1=11 (line number that contains "t")
TARGET.2=6 (column number that contains "t")
TARGET.3=11 (line number that contains "y")
TARGET.4=8 (column number that contains "y")
```

These values are returned because "try" is the last string found in the target.

Information returned by EXTRACT/TARGET/ is only guaranteed to be valid when the EXTRACT is done immediately following the LOCATE or CLOCATE of the target. Any XEDIT subcommand issued between the LOCATE or CLOCATE of the target and the EXTRACT has the potential to invalidate the TARGET information.

```
     TARGET.0      number of variables returned
           .1      line number of first character
           .2      column number of first character
           .3      line number of last character
           .4      column number of last character
```

TERMinal

returns "DISPLAY" or "TYPEWRITER" as defined in the SET TERMINAL subcommand.

```
   TERMINAL.0      number of variables returned
           .1      DISPLAY|TYPEWRITER
```

TEXT

returns "ON" or "OFF" as specified in the SET TEXT subcommand.

```
       TEXT.0      number of variables returned
           .1      ON|OFF
```

TOF

returns "ON" or "OFF" as determined by the editor. TOF is "ON" when the line pointer reaches the Top of File (or Top of Range) line.

```
        TOF.0      number of variables returned
           .1      ON|OFF
```

TOFEOF

returns "ON" or "OFF" as specified in the SET TOFEOF subcommand.

```
     TOFEOF.0      number of variables returned
           .1      ON|OFF
```

TOL

returns "ON" or "OFF" as determined by the editor. TOL is "ON" when the column pointer reaches zone1-1.

```
        TOL.0      number of variables returned
           .1      ON|OFF
```

TRANSLat

returns "ON" or "OFF", depending on whether or not the user has defined pairs of uppercase translate characters using the SET TRANSLAT subcommand.

```
   TRANSLAT.0      number of variables returned
           .1      ON|OFF
```

TRunc

returns the truncation column number as defined by the SET TRUNC subcommand.

```
      TRUNC.0      number of variables returned
           .1      truncation column number
```

UNIQueid

returns the unique identifier associated with the file. The identifier has the form *rrrnnnnn* where *rrr* is the number of times XEDIT was called recursively and *nnnnn* is the current autosave number. Note that when the recursion level, *rrr*, is less than 100, the leading zero(es) will be dropped. The uniqueid is also used as the file name for the AUTOSAVE file.

```
UNIQUEID.0      number of variables returned
        .1      unique identifier associated
                 with this file
```

UNTil

returns the file type up through which updates were applied as specified on the XEDIT command or LOAD subcommand.

```
   UNTIL.0      number of variables returned
        .1      file type(if specified) or blanks
```

UPDate

returns "ON" or "OFF" as determined by the editor. Update is "ON" when the XEDIT command or LOAD subcommand has been issued and the UPDATE option was specified or implied.

```
  UPDATE.0      number of variables returned
        .1      ON|OFF
```

VARblank

returns "ON" or "OFF" as specified in the SET VARBLANK subcommand.

```
VARBLANK.0      number of variables returned
        .1      ON|OFF
```

Verify

returns the verification columns and "ON" or "OFF" as specified in the SET VERIFY subcommand.

```
  VERIFY.0      number of variables returned
        .1      ON|OFF
        .2      columns
```

VERShift

returns n or -n, which is the relative position of the screen over the file, as a result of any LEFT or RIGHT subcommands.

```
VERSHIFT.0      number of variables returned
        .1      n|-n
```

Width

returns the WIDTH value specified in the XEDIT command or LOAD subcommand.

```
   WIDTH.0      number of variables returned
        .1      width of file
```

WINdow

is the name of the virtual screen and window that XEDIT will use to display the file or files being edited.

```
    WINDOW.0  number of variables returned
          .1  window name or blanks (for non-display
               terminals)
```

WRap

returns "ON" or "OFF" as specified in the SET WRAP subcommand.

```
    WRAP.0      number of variables returned
        .1      ON|OFF
```

Zone

returns the left and right zone column numbers specified in the SET ZONE subcommand.

```
        ZONE.0      number of variables returned
            .1      left zone
            .2      right zone
```

=

returns the string in the equal(=) buffer. The = buffer contains the last executed subcommand, macro, CP/CMS command, or whatever has been specified in the SET = subcommand.

```
   EQUALSIGN.0      number of variables returned
            .1      the string in the = buffer
```

Notes for Macro Writers

1. The number of variables returned by EXTRACT may depend upon whether the setting is "ON" or "OFF" or whether the settings were requested on a typewriter terminal. The following settings return a value of "name.0=0" on a typewriter terminal. No other variables associated with that setting will be initialized:

```
CMDLINE     LSCREEN     SCREEN
CURSOR      MSGLINE     TABLINE
FLSCREEN    SCALE
```

2. If an error occurs while processing the request specified on the EXTRACT subcommand, a nonzero return code is returned and an error variable, EXTRACT.n is set. EXTRACT.0 and EXTRACT.1 are set on return codes 2, 5 and 16 only. EXTRACT.0 is always set to the number of error variables that are returned. If the return code is 2 or 5, then EXTRACT.1 is set to the delimited string which was invalid (return code 5) or the delimited string containing a target that was not found (return code 2) when EXTRACT was invoked. All values prior to the one specified in EXTRACT.1 will have been processed, while those following the one in error will not.

   However, when the return code is 16, EXTRACT.1 is set to the name of the variable which was too long for the EXEC 2 restriction of a maximum value length of 255 characters. For example, if one is editing a file with a current line that is longer than 255 characters, and an EXTRACT /CURLINE/ is issued, then EXTRACT.1 will be set to "CURLINE.3". The setting of any other variables resulting from that invocation of EXTRACT is unpredictable. EXTRACT.1 does not give an indication of what has or has not been set in this case.

3. If EXTRACT is issued when an exec or XEDIT macro is not active, a return code of −3 will be set and the following error message is displayed:

```
631E EXTRACT can only be issued from an EXEC 2 or REXX exec.
```

   If XEDIT is invoked from an EXEC or XEDIT macro, then that EXEC or macro is "active" until you exit from XEDIT and subsequently from the EXEC or macro. Issuing EXTRACT from the XEDIT command line may set variables in the EXEC or macro that invoked XEDIT.

4. EXTRACT/MASK/issued from an EXEC 2 environment will always result in return code 16 since the MASK has 256 characters. To obtain the same information, you may use TRANSFER MASK. EXTRACT/MASK/ issued from a REXX environment will return the mask properly, as REXX imposes no length restrictions.

5. If SET HEX ON is in effect, targets can be specified in hexadecimal.

Messages

545E	Missing operand(s) [RC=5]
622E	Insufficient [free] storage [*message*] [RC=104]
631E	*function* can only be executed from an EXEC-2 or REXX EXEC [or as a CMS command] [RC=-3]

Return Codes

−3	Invalid when issued from an environment other than EXEC 2 or REXX
2	Target not found
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
16	EXEC 2 variable greater than 255 characters
104	No storage is available

Examples

1. If you edit a new file with a default record length of 80 and issue:

   ```
   set lrecl 65
   ```

   and then execute an XEDIT macro which issues:

   ```
   extract /ACTION/
   ```

   ```
   The following variables are set:  ACTION.0 = 1
                                      ACTION.1 = ON
   ```

2. If you edit a new file SAMPLE FILE A and then execute an XEDIT macro which issues:

   ```
   extract ?COL?fname?pf3?
   ```

   ```
   The following variables are set: COLUMN.0 = 1
                                     COLUMN.1 = 1
                                     FNAME.0  = 1
                                     FNAME.1  = SAMPLE
                                     PF3.0    = 2
                                     PF3.1    = BEFORE
                                     PF3.2    = QUIT
   ```

3. While editing a file you make line 6 the current line. There are no entries in the pending list. You then execute an XEDIT macro which issues:

   ```
   set pending block 3WW57
   extract !pending block * :2 +5!
   ```

   ```
   The following variables are set: PENDING.0 = 7
                                     PENDING.1 = 6
                                     PENDING.2 = WW
                                     PENDING.3 = WW
                                     PENDING.4 = BLOCK
                                     PENDING.5 = 3
                                     PENDING.6 = 57
                                     PENDING.7 = null
   ```

4. While editing a file you execute an XEDIT macro which issues:

   ```
   set case mixed respect
   extract ¢CASE¢COLOR ALL¢AUTOSAVE¢
   ```

   ```
   The following variables are set: CASE.0 = 2
                                    CASE.1 = MIXED
                                    CASE.2 = RESPECT
                                 EXTRACT.0 = 1
                                 EXTRACT.1 = COLOR ALL
   ```

   Note: The return code is 5 because COLOR ALL is an invalid operand and nothing is returned for AUTOSAVE (see notes for macro writers, 2).

5. While editing a file on a typewriter terminal, you insert a line containing the following text:

   ```
   Professor Twist could not but smile.
   ```

   With the line you just added being the current line, you then execute an XEDIT macro which issues:

   ```
       extract %SCALE%CURLINE%
   ```

   ```
   The following variables are set:

            SCALE.0 = 0
          CURLINE.0 = 4
          CURLINE.1 = -1
          CURLINE.2 = -1
          CURLINE.3 = Professor Twist could not but smile.
          CURLINE.4 = ON
          CURLINE.5 = NEW
   ```

FILE

Use the FILE subcommand to write the edited file on disk or in an SFS directory. You may also use the FILE subcommand to override the file identifier originally supplied in the XEDIT command.

Format

FILE	$\left[\begin{matrix} fn \\ = \end{matrix} \left[\begin{matrix} ft \\ = \end{matrix} \left[\begin{matrix} fm \\ = \end{matrix}\right]\right]\right]$

Operands

fn
: indicates the file name for the file (or membername when editing a member of a MACLIB). If you do not specify fn (or =), ft and fm cannot be specified, and the existing file name, file type, and file mode are used.

ft
: indicates the file type for the file.

fm
: is the file mode of the file to be written.

Usage Notes

1. If the file mode indicates an accessed minidisk, the minidisk must be accessed read/write. If the file mode indicates an accessed SFS directory, it can be accessed as either read/only or read/write.
2. You must have write authority to an existing file to file it in another user's SFS directory.
3. If you have write authority to another user's SFS directory, you can create a new file in that directory even though you have the directory accessed read/only. The owner of the directory is now the owner of the new file, but because you created the file, you have write authority for the file.
4. You can change the file name, file type, and file mode during an editing session by using the SET FNAME, SET FTYPE, and SET FMODE subcommands (as well as specifying a different file identifier on the FILE subcommand).
5. If you change the file identifier (to a unique file identifier) and the file being edited had been previously written to disk or directory, that copy with the original name of the file is not altered.
6. If you try to FILE a shared file that has been modified by another user during your editing session, the editor stops the FILE operation and displays the following message:

   ```
   594E File fn ft fm already exists or changed; use FFILE or SSAVE
   ```

   At this point you can decide whether to preserve the other user's changes or write over them. If you want to write over the other user's changes, causing them to be lost, enter FFILE (abbreviated as FF). If not, change the file identifier so it is unique and reissue the FILE subcommand to save your changes under a different name. Later, you can determine what the differences are between the two versions of the file and combine them.

To avoid this situation, use the LOCK option on the XEDIT command to prevent other users from changing the file while you are editing it. The LOCK option is the default.

Message 594E is also displayed if you change the file identifier so that it is identical to that of an *existing* file and try to file it.

This "protected FILE" works in the following way. FILE is defined as a synonym to PFILE, which is the protected equivalent of FILE. FFILE is a synonym for COMMAND FILE.

7. An equal sign (=) may be used for any operand. Using an equal sign means that the corresponding value of the current file is to be used.

8. If the FILE subcommand is issued for a file whose temporary copy in virtual storage has no records (that is, it is empty), the permanent copy of the file on disk or directory is not changed. The following message is displayed and the editor is exited:

```
559W Warning: empty file not written to disk
```

9. Changing a membername

   If you change the membername (to a unique membername) and the member has previously been written to a disk or directory, that copy with the original name of the member is not altered.

   However, if you change the membername while editing a member so that the new membername is identical to that of an existing member in that library, the editor stops the FILE operation and displays the following warning:

```
594E File fn ft fm already exists or changed; use FFILE or SSAVE
```

   If you want to write over the existing member, enter FFILE (abbreviated as FF). If not, change the membername so that it is unique and reissue the FILE subcommand.

10. Members of a MACLIB must have a file type of MEMBER.

11. Members with identical names contained in libraries with identical names, but residing on different disks or directories, will not be written over if you attempt to change the file mode. The following warning will be issued:

```
594E File fn ft fm already exists or changed; use FFILE or SSAVE
```

   For example:

```
PROJA MACLIB A1          PROJA MACLIB B1

ENTER                    ENTER
LEAVE                    LEAVE
FORMAT                   FORMAT
COMPUTE                  COMPUTE
```

   If you are editing the ENTER member in the PROJA MACLIB that resides on your file mode B and attempt to change the file mode by issuing "FILE = = A", the warning message is issued.

12. When editing two or more members of a MACLIB in member mode, the lock on the library is not deleted until the last member that had the lock option in effect exits the ring.

Notes for Macro Writers

1. If FILE is issued from a macro, the file is written to disk or directory, but control remains in the macro. When the macro finishes executing, control is returned to the editor.

   If multiple files were being edited, only the current file is written to disk or directory and removed from the set of files being edited.

   If only one file was being edited, a FILE issued from a macro sets the return code to 1 and executes the FILE when the macro completes execution. After issuing the FILE, any subcommands issued in the macro will result in a return code of 6.

2. You cannot issue FILE, FFILE or PFILE from a prefix macro. However, if you issue a FILE, which consists of a SAVE and a QUIT, the SAVE will be performed and you will receive error message 509E on the QUIT.

Responses

If only one file was edited, the CMS ready message indicates that the file has been written to disk or directory. The user is returned to the environment that invoked the editor.

If more than one file was being edited, the current file is written to disk or directory and is removed from the set of files being edited.

On a typewriter terminal, the following message is displayed:

```
553I Editing file: fn ft fm
```

Messages

002E	File *fn ft fm* not found [RC=28]
007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
033E	File *fn ft fm* is not a library [RC=32]
037E	Filemode *mode* is accessed as read/only [RC=12]
039E	No entries in library *fn ft fm* [RC=32]
048E	Invalid mode *mode* [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
069E	Filemode *mode* not accessed [RC=36]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, 99, or 100]
105S	Error *nn* writing file *fn ft fm* on disk [RC=55, 70, 76, 99, or 100]
157S	MACLIB limit exceeded [RC=88]
167S	Previous MACLIB function not finished [RC=88]
229E	Unsupported OS dataset, error *nn* [RC=80, 81, 82, 83, or 84]
509E	*subcommand* subcommand not valid from a prefix macro [RC=4]
520E	Invalid operand: *operand* [RC=5]
531E	Disk or file space is full; set new filemode or clear some space [RC=13]
532E	Disk or file space is full; AUTOSAVE failed
554E	Not enough virtual storage available [RC=104]
559W	Warning: empty file not written to disk
560E	Not enough space for serialization between TRUNC and LRECL

579E	Records truncated to *nn* when added to *fn ft fm* [RC=3]
594E	File *fn ft fm* already exists or changed; use FFILE or SSAVE [RC=3]
598S	Unable to build update file: internal list destroyed [RC=7]
599S	Unable to build update file: serialization destroyed [RC=7]
622E	Insufficient free storage for reading map [RC=104]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1141W	User filespace threshold exceeded
1215E	File *fn ft fm* is locked by another user [RC=70]
1258E	Not authorized to write file *fn ft fm* [RC=12]
1262S	Error *nn* opening file *fn ft fm* [RC=55, 70, 76, 99, or 100]
1301S	Rollback error *nn*, file *fn ft fm* left open

Return Codes

0	Normal
1	File has been filed and was the only one edited
3	File already exists; truncated or spilled
4	Invalid when issued from a prefix macro
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
7	Error building the update file
12	Minidisk defined in file mode is read-only; or not authorized to write to file
13	Minidisk or file space is full
20	Invalid character in file name or file type
24	Invalid file mode
28	File not found
31	A rollback occurred
32	File is not a library, or library has no entries, or file is not fixed, 80 char. records
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
84	Unsupported OS data set
88	Previous Maclib function not finished, or Maclib limit exceeded
99	A required system resource is not available
100	Error 'nn' reading|writing file from|to disk
104	Insufficient virtual storage

FIND

Use the FIND subcommand to search forward in the file for the first line that starts with the text specified in the operand. Only the non-blank characters in the operand are checked against the file. The search starts with the line following the current line.

Format

Find	*text*

Operand

text
is any text that you expect to find, beginning in column one of the next file line. If text contains imbedded blanks, those character positions in the file line are not checked.

Usage Notes

1. If the SET IMAGE ON subcommand has been issued, tab characters are converted to blanks (or filler characters) before the search is made. In addition, the search starts in the first tab column.
2. Use an underscore character (_) in the operand to specify that a blank character should be present in a line.
3. Only one blank can be used as a delimiter following the FIND subcommand; the operand starts after the blank.
4. See also FINDUP, NFIND, and NFINDUP.
5. If the subcommand SET WRAP OFF has been issued, the search will continue until the end of file.

   If the subcommand SET WRAP ON has been issued, the search will continue through the entire file and stop when the line where it started is reached again. When a wrap occurs under these conditions, the following warning message is displayed:

```
592W Wrapped ....
```

6. The FIND subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Response

If text is found, the line containing the specified text becomes the new current line.

Messages

545E	Missing operand(s) [RC=5]
586E	Not found [RC=2]
592W	Wrapped ....

Return Codes

0	Normal
2	No line was found
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== Dingoes are wild dogs of Australia.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====          .
=====          .
===== They are called Dingoes.
=====          .
=====          .
=====          .
===== Dingoes do not bark.
```

`find Ding` (text must start in column one of a line)

New Current Line:

```
===== Dingoes do not bark.
```

FINDUP

Use the FINDUP subcommand to search *backward* in the file for the first line that starts with the text specified in the operand. Only the non-blank characters in the operand are checked against the file. The search starts with the line preceding the current line.

Format

FINDUp **FUp**	*text*

Operand

text
is any text that you expect to find beginning in column one of a file line that appears before the current line. If text contains imbedded blanks, the corresponding character positions in the file are not checked.

Usage Notes

1. If the SET IMAGE ON subcommand has been issued, tab characters are converted to blanks (or filler characters) before the search is made. In addition, the search starts in the first tab column.
2. Use an underscore character (_) in the operand to specify that a blank character should be present in a line.
3. Only one blank can be used as a delimiter following the subcommand; the operand starts after the blank.
4. See also FIND, NFIND, and NFINDUP.
5. If the subcommand SET WRAP OFF has been issued, the search will continue until the top of file.

   If the subcommand SET WRAP ON has been issued, the search will continue through the entire file and stop when the line where it started is reached again. When a wrap occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```

6. The FINDUP subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Response

If text is found, the line containing the specified text becomes the new current line.

Messages

545E	Missing operand(s) [RC = 5]
586E	Not found [RC = 2]
592W	Wrapped ....

Return Codes

0	Normal
2	No line was found
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

FORWARD

Use the FORWARD subcommand to scroll toward the end of a file for a specified number of screen displays.

Format

FOrward	[*n* | * | **1**]

Operand

n

is the number of screen displays you want to scroll forward. If you specify an asterisk, the line pointer moves to the "End of File" line. If you omit *n*, the screen scrolls forward for one display.

Usage Notes

1. The FORWARD subcommand is assigned by the editor to the PF8 key.
2. Issuing `FORWARD 0` from anywhere in the file makes the first line of the file the new current line.

Responses

If you issue the FORWARD subcommand when the current line is the last line of the file, the End of File line becomes the new current line. If you issue the FORWARD subcommand when the current line is the End of File line, the editor "wraps around" the file, making the first line of the file the new current line.

Messages

520E	Invalid operand: *operand* [RC=5]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
543E	Invalid number: *number* [RC=5]

Return Codes

0	Normal
1	End of file reached (subsequent FORWARD restarts at top of file)
3	Terminal is not a display
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

GET

Use the GET subcommand to insert all or part of a specified CMS file after the current line of the file that you are editing. You can also use the GET subcommand *without operands* to insert lines saved by a previous PUT or PUTD subcommand.

Format

GET	$\left[\begin{matrix}fn\\=\end{matrix}\left[\begin{matrix}ft\\=\end{matrix}\left[\begin{matrix}fm\\=\end{matrix}\right]\left[\begin{matrix}firstrec\\\underline{1}\end{matrix}\quad\left[\begin{matrix}numrec\\\underline{*}\end{matrix}\right]\right]\right]\right]$

Operands

fn
: is the file name of the file that contains the data to be inserted into the file you are editing.

ft
: is the file type of the file that contains the data to be inserted. If ft is not specified, the file type of the file you are editing is assumed.

fm
: is the file mode of the file that contains the data to be inserted. If you do not specify fm, all of your accessed modes are searched for the file.

firstrec
: indicates the record number of the first record you want to insert. If firstrec is not specified, the first record in the file is the default.

numrec
: indicates the number of lines to be inserted, starting with the line specified by firstrec. If numrec is not specified, or is specified as *, then the remainder of the file between firstrec and the end of the file is inserted.

Usage Notes

1. The GET operand list is positional; if you omit one operand *except* for file mode, which is optional, you cannot specify any operands that follow. Therefore, if you want to specify firstrec and numrec, you must specify the file name and file type of the file.
2. If you do not specify firstrec and numrec, the editor inserts the entire file.
3. If the record length of the records in the file containing the data to be inserted exceeds that of the file being edited, records are truncated and a message is displayed, unless SET SPILL ON or SET SPILL WORD has been issued. In that case, a message is displayed indicating that the text has been spilled, and the characters beyond the record length are inserted in the file as one or more lines, starting with the first character or word that would have gone beyond the record length. If the record length is shorter, the records are padded with blanks to the record length of the file being edited and inserted in the file.
4. If the editor fills up available storage while executing a GET request, it may not be able to copy all of the file.
5. The operands fn, ft, or fm may be specified using an equal sign (=), in which case the corresponding value in the file being edited is used.

6. If you issue a GET subcommand for a file that is in packed format, the editor does not unpack the file.

Responses

The last line inserted becomes the new current line.

When the end of the file that is being inserted is reached, the following message is displayed:

```
564W  EOF reached
```

Messages

002E	File *fn ft fm* not found [RC=28]
048E	Invalid filemode *mode* [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
069E	Filemode *mode* not accessed [RC=36]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, or 100]
156E	Record *nnn* not found--file *fn ft fm* has only *nnn* records [RC=32]
229E	Unsupported OS data set, error *nn* [RC=80, 81, 82, 83, or 84]
520E	Invalid operand: *operand* [RC=5]
521E	Invalid line number [RC=5]
543E	Invalid number: *number* [RC=5]
554E	Not enough virtual storage available [RC=104]
557S	No more storage to insert lines [RC=4]
562E	No line(s) saved by PUT(D) subcommand [RC=28]
563W	Records {truncated|spilled} [RC=3]
564W	EOF reached
565W	EOF reached; records {truncated|spilled} [RC=3]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 99, or 100]

Return Codes

0	Normal
3	Truncated or spilled
4	No more storage available to insert lines
5	Invalid operand or (line) number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
20	Invalid character in file name or file type
24	Invalid file mode
28	Source file not found
31	A rollback occurred
32	Record *firstrec* is beyond end of file
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.

83	The OS data set or DOS file has more than 16 user labels or data extents.
84	Unsupported OS data set
99	A required system resource is not available
100	Error reading file from disk
104	Insufficient storage available

Examples

The following are valid ways to issue a GET subcommand:

```
get
```

(insert all lines previously stored by a PUT or PUTD)

```
get file2 data
```

(insert the entire file, FILE2 DATA, after current line of this file)

```
get file2 data 1 10
```

(insert the first ten lines of FILE2 DATA)

The following sequence illustrates how to use GET and PUT to transfer data between files, when editing multiple files:

1. `xedit file1 mine`

   This file appears on the screen.

   Position the line pointer at the line after which you want to insert lines.

2. `xedit file2 data`

   This file now appears on the screen.

   The status area indicates that you are editing two files.

   Move the line pointer to the beginning of lines to be inserted, using, for example, CLOCATE, UP, NEXT, etc.

3. `put target`

   Stores lines from current line up to the target line.

4. `quit`

   The first file, FILE1 MINE, reappears on the screen.

5. `get` (no operands)

   The lines are inserted.

HELP (Macro)

Use the HELP macro to get online information about XEDIT subcommands, macros, and messages or to invoke the CMS HELP command. The XEDIT HELP facility allows you to display an individual XEDIT HELP file, a menu that lists all the XEDIT HELP files, or a series of menus that list XEDIT HELP files organized by various editing tasks. HELP files contain brief information, detailed descriptions, formats, parameters, options, usage notes, and related information for the XEDIT subcommands and macros.

Format

Help	[|<u>**MENU**</u> |**HELP** |**TASK**| *name*]

Operands

<u>MENU</u>
displays a list of all XEDIT subcommand and macro HELP files.

HELP
displays how to use the XEDIT macro HELP.

TASK
displays a list of some task-oriented HELP files.

name
can be any XEDIT subcommand name or XEDIT macro name. Specifying a name causes a description of the subcommand or macro to be displayed. If name is not an XEDIT subcommand or macro, the entire HELP subcommand is transferred to the CMS HELP command, where it must follow the CMS HELP command syntax (refer to the *VM/SP CMS Command Reference*).

Usage Notes

1. XEDIT subcommand or macro abbreviations can be used in the name operand.
2. Use the CMS HELP command format to display information on CMS as well as XEDIT error messages. For example:

   ```
   help dmsnnnnt or help dmsxxxnnnnt
   ```

   where *xxx* is the module name, *nnnn* is the message number, and *t* is the message type.
3. HELP MENU is initially set to the PF1/PF13 key.
4. Task menus are available in the RELATED sections of the SET, QUERY, and PREFIX commands. These RELATED task menus list and briefly describe the SET, QUERY, and PREFIX operands available in XEDIT.
5. Error message text information has been replaced with a reference to the proper manual for information. But you can still use the ERRORS option to tailor your own HELP files. For information on tailoring your own HELP files, see the *CMS User's Guide*.

6. Although the RELATED option is available, few examples of RELATED information files will appear in the HELP files. However, you can use this option to tailor your own HELP files. For information on tailoring your own HELP files to include this option, see the *CMS User's Guide*.

7. If you are viewing a command HELP file, you can issue the MOREHELP command from the command line. For example, if you are viewing the HELP file for the XEDIT CHANGE command and then decide you want to see the format section for that command, you can enter the following on the command line:

   ```
   MOREHELP (FORMAT
   ```

   The format section of the CHANGE command will then be displayed.

Messages

From the CMS HELP facility.

Return Codes

0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
any number > 10	Standard CMS HELP command return codes

HEXTYPE (Macro)

Use the HEXTYPE macro to display a specified number of lines in both hexadecimal and EBCDIC.

Format

HEXType	[*target* |1]

Operand

target
defines the number of lines to be displayed in both hexadecimal and EBCDIC. The display starts with the current line and goes up to but does not include the target line. If you specify an asterisk (*), the rest of the file is displayed both ways. If you omit target, only the current line is displayed both ways.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Response

Each line specified is displayed first in hexadecimal and then in EBCDIC.

Only the data within the first pair of verify columns is displayed.

The line pointer moves to the last line typed.

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]
581E	Subcommand is not valid in extended mode [RC=3]

Return Codes

0	Normal
1	TOF or EOF reached
2	Target line not found
3	Subcommand is not valid in extended mode
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== To be or not to be -
```

`hextype` (display the current line in hexadecimal and character)

```
E3964082 85409699 409596A3 40A39640 82854060
 T o  b e  o r  n o t  t o  b e -
```

INPUT

Use the INPUT subcommand to insert a single line into a file, or, if no data line is specified, to leave edit mode and enter input mode.

Format

Input	[*line*]

Operand

line

is the input line to be entered into the file. It can contain blanks and tabs, which are converted to blanks if the SET IMAGE ON subcommand has been issued. The line is inserted after the current line, with the first character in the first tab column (only if SET IMAGE ON is in effect) and becomes the new current line. If you enter at least two blanks following the INPUT subcommand (and no additional text), a blank line is inserted into the file.

The preceding description applies when the INPUT subcommand is entered in the form `INPUT line`. Input mode (INPUT entered with no operand) is described below.

Usage Notes

1. When you issue the INPUT subcommand without an operand, the screen display changes in the following ways:

   a. The phrase "573I Input mode:" appears in the message area, and "Input-mode n File(s)" appears in the status area of the screen.

   b. The phrase * * * Input Zone * * * appears in the command line.

   c. The prefix area disappears from the display.

   d. All file lines following the current line disappear from the display. Lines pre-filled with blanks appear in their place. (You can use the SET MASK subcommand to fill this area with something other than blanks.) This blank area, between the current line and the command line, is the input zone.

   e. The cursor is placed on the first line of the input zone, which is the blank line immediately following the current line. You can then type in new lines of data in the input zone.

   Only data typed in the columns defined by the SET VERIFY subcommand and up through the truncation column (defined by the SET TRUNC subcommand) is accepted. Data that is entered beyond the truncation column or outside the current SET VERIFY settings is lost.

2. When you fill up the screen but wish to stay in input mode and type in more lines, press the ENTER key once. The lines that you typed move to the top half of the screen, with the last line you typed becoming the new current line; it is followed by blank lines. If AUTOSAVE is set for your editing session, you may receive the message, `510I Autosaved as 'fn ft fm'`, depending on the value set on the SET AUTOSAVE subcommand.

3. When you are finished typing in data and want to return to edit mode, press the ENTER key twice. The last line you typed in input mode becomes the current line, the screen layout is restored, the phrase "587I XEDIT: " appears in the message area, and "X E D I T n File(s)" appears in the status area of the screen. (If you did not type in new lines while in input mode, you can return to edit mode by pressing the ENTER key once.)
4. You can vary the size of the input zone by using the SET CURLINE subcommand to change which line on the screen is the current line. For a larger input zone, move the current line to the top of the screen; for a smaller input zone, move the current line lower on the screen.

   If you move the current line to the last available line in the file area, you cannot enter data unless you also move the command line (by using the SET CMDLINE subcommand). In general, you should leave space between the current line and the bottom of the screen for input mode.
5. If you issue an INPUT subcommand when the current line is the End of File line, the lines are inserted after the last file line.
6. When you use PF or PA keys in input mode, the editor automatically exits from input mode, enters edit mode to execute the subcommand associated with the PF or PA key, and returns to input mode. Therefore, you should carefully select which PF or PA keys you use in input mode. For example, if you press a PF key assigned to the FORWARD subcommand, the editor scrolls the screen forward and then returns you to input mode, but the input area will be on the next screen. In addition, some PF or PA keys are meaningless when used in input mode, for example, a PF key assigned to the ? subcommand.

   PF or PA keys that are particularly useful in input mode are those assigned to TABKEY, SPLTJOIN (or SPLIT and JOIN), NULLKEY, and RGTLEFT.
7. On a typewriter terminal:
   a. Pressing the RETURN key causes any line that is typed in to be inserted into the copy of the file that is kept in storage.
   b. If the SET IMAGE ON subcommand has been issued, tabs are converted to blanks before a line is inserted into the file.
   c. The editor interprets a line that has an escape character in column 1 (see the SET ESCAPE subcommand) as an XEDIT subcommand. The subcommand is executed and input mode is reentered automatically. (You cannot use SET ESCAPE on a terminal in DISPLAY mode.)
   d. Enter a null line to leave input mode and return to edit mode.
8. If SET HEX ON is in effect, the line can be specified in hexadecimal. For example, INPUT X'C1C2C3'.
9. When the INPUT subcommand is entered with text specified (that is, in the form "INPUT line"), and SET SPILL OFF is in effect (the default), characters entered beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters entered beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.

   When the INPUT subcommand is entered with no text specified (that is, in the form "INPUT") and input mode is entered, data is handled as described in usage note 1, above.

10. If you enter the LINEND symbol while in input mode and press the ENTER key, your line will not be entered as separate lines in the file. Instead, it will be shown as a string with the LINEND symbol appearing literally.
11. When you are at the end of file (or end of range), and you enter the INPUT subcommand, the last line of the file (or range) is displayed as the current line, even if that line is not within the defined scope.
12. If multiple logical screens have been defined by the SET SCREEN subcommand, then entering input mode causes subcommands entered on other logical screens to either
    a. be ignored, if the screen contains a view of the same file, or
    b. remain on the command line and not execute until input mode is exited, if the screen contains a view of a different file.

Messages

503E	{Truncated|Spilled} [RC=3]
557S	No more storage to insert lines [RC=4]
614E	Screen modifications lost. See SET FULLREAD ON to use PA keys safely [RC=8]

Return Codes

0	Normal
3	Truncated or spilled
4	No more space available to add lines
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Modifications lost because PA key pressed when message pending

JOIN (Macro)

Use the JOIN macro to combine two or more lines into one replacement line.

The first format enables you to join two lines at the column pointer or at the cursor.

The second format enables you to join two or more lines at a specified column number(s) or to insert a specified character string(s) before appending the next line.

In all cases, the lines that are appended (the original lines) are deleted. However, if data precedes zone1 (see SET ZONE), only the data following zone 1 is deleted.

Format

Join	**[ALigned]**	**Column** **CURSOR**
	[ALigned]	*colno* . . . */string/*

Operands

no operand
joins the current line and the next line. The next line is appended after the first trailing blank in the current line.

ALigned
removes up to the same number of leading blanks from the line being joined as there are on the line to which it is appended.

Column
joins the current line and the next line, which overlays the current line starting at the column pointer. The line pointer and the column pointer remain unchanged.

CURSOR
joins the line containing the cursor and the next line, which overlays the line starting at the cursor position. JOIN CURSOR is most useful when assigned to a PF key.

colno
specifies a column number in the current line where the next line is to be appended. Subsequent consecutive lines are appended to (and overlay the contents of) the current line for as many times as there are column numbers (colno) specified.

/string/
inserts the specified string (without delimiters) in the current line, starting at the first trailing blank location. Then the next line is appended after the string. This process is repeated for as many times as there are /string/ operands specified.

Usage Notes

1. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. For the JOIN macro, SET SPILL OFF (the default) has the same effect as SET SPILL ON. JOIN does not truncate data.
2. The original lines that are appended as a result of a JOIN macro are deleted, unless data precedes zone 1 (see SET ZONE). In this case only the data following zone 1 is deleted.
3. Before using JOIN COLUMN, check to see if the column pointer is in the desired location, because the next line is appended starting at the column pointer. Use the CLOCATE subcommand to move the column pointer, if necessary.
4. The line pointer and column pointer remain unchanged.
5. JOIN is the converse of SPLIT. See also SPLTJOIN, which combines SPLIT and JOIN.
6. The cursor or the column number or string specified must fall within the current zones.

Messages

503E	{Truncated|Spilled} [RC=3]
520E	Invalid operand: *operand* [RC=5]
526E	Option *option* valid in display mode only [RC=3]
561E	Cursor is not on a valid data field [RC=1]
564W	EOF reached [RC=1]
575E	Invalid [argument or]{JOIN|SPLIT|TABS|VERIFY|ZONE} columns defined [RC=5]
585E	No line(s) changed [RC=1]
685E	Joined lines(s) exceed zone settings [RC=5]

Return Codes

0	Normal
1	No line(s) changed, or cursor is not on a valid data field
3	Spilled, or operand is valid only for display terminal
5	Invalid operand, or joined line(s) exceed zone settings
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

If you assign PF11 to JOIN CURSOR:

```
set pf11 join cursor

===== This line is _
===== too short.
```

Note position of the cursor (_) in the first line. Pressing the PF11 key produces the following line:

```
===== This line is too short.
```

Using JOIN to Insert a Character String:

```
===== .sp
===== .in 5
===== .of 3
```

`join /;/ /;/` (join the current line and the next two, separated by semi-colons)

```
===== .sp;.in 5;.of 3
```

Using JOIN ALIGNED to remove up to the same number of leading blanks from the line being joined to the current line:

```
=====      These lines have _
=====      leading blanks.

join al cursor

=====      These lines have leading blanks.
```

Using JOIN to Join Lines Separated by Blanks:

```
===== Electric eels
===== can discharge bursts
===== of 625 volts.
```

`join / / / /` (join current line and next two, separated by blanks)

```
===== Electric eels can discharge bursts of 625 volts.
```

LEFT

Use the LEFT subcommand to view columns of data that are not currently visible on the screen. The LEFT subcommand allows you to see data that is *to the left of the first column* on the screen. The data moves to the right, thus allowing you to see a specified number of positions to the left of the first column.

Format

LEft	[*n* |1]

Operand

n

specifies the number of positions to the left of the first column on the screen that you want to see. If you omit n, one position to the left of the first column becomes visible.

Usage Notes

1. The LEFT subcommand does not cause data to be lost, nor does it move the line or column pointer.
2. To get the data back to its original position, use the RIGHT *n* subcommand. See also the RGTLEFT macro in this publication.
3. LEFT subcommands are cumulative. For example,

   ```
   left    10
   left    10
   ```

   is equivalent to

   ```
   left    20
   ```

   Therefore, if several LEFT subcommands have been issued, column one on the screen might not contain the first character in a line.
4. If you have issued several LEFT and/or RIGHT subcommands and have forgotten the value of n:
   a. LEFT 0 or RIGHT 0 restores the screen to its original display.
   b. SET VERIFY resets LEFT (or RIGHT) to zero.
   c. QUERY VERSHIFT displays −n (result of a LEFT) or n (result of a RIGHT).
5. The total number of columns specified cannot exceed the logical record length.

Notes for Macro Writers

EXTRACT /VERSHIFT/ returns the value of n or minus n.

Response

The screen moves to the left relative to the file data.

Messages

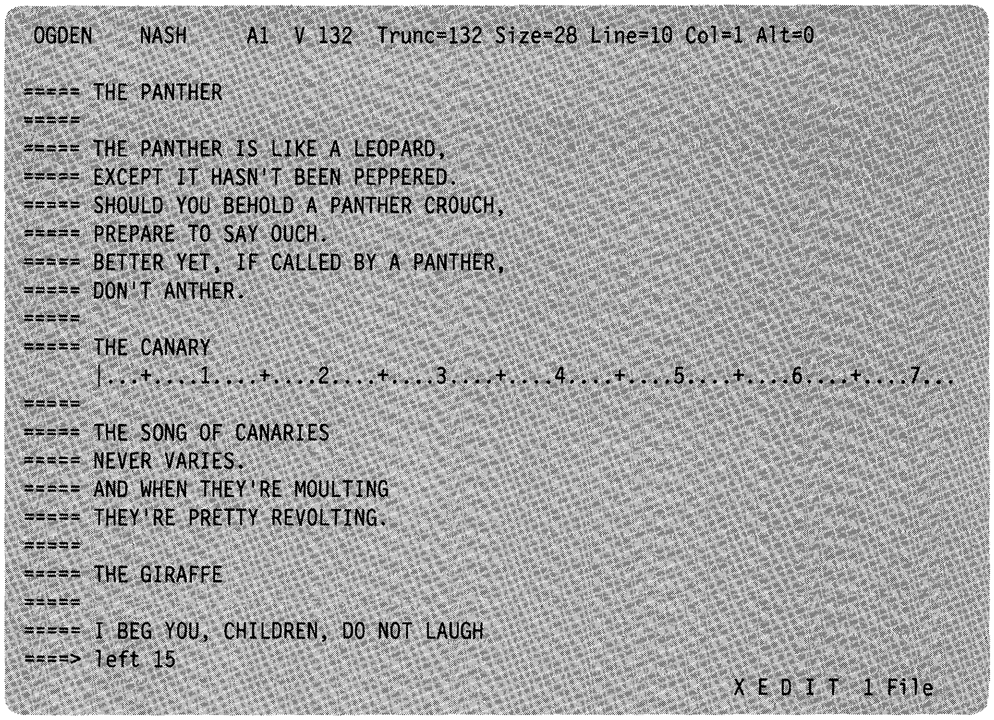
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
576E	{Total verify width exceeds screen size (*nn*)|Total offset exceeds LRECL (*nn*)} [RC=5]

Return Codes

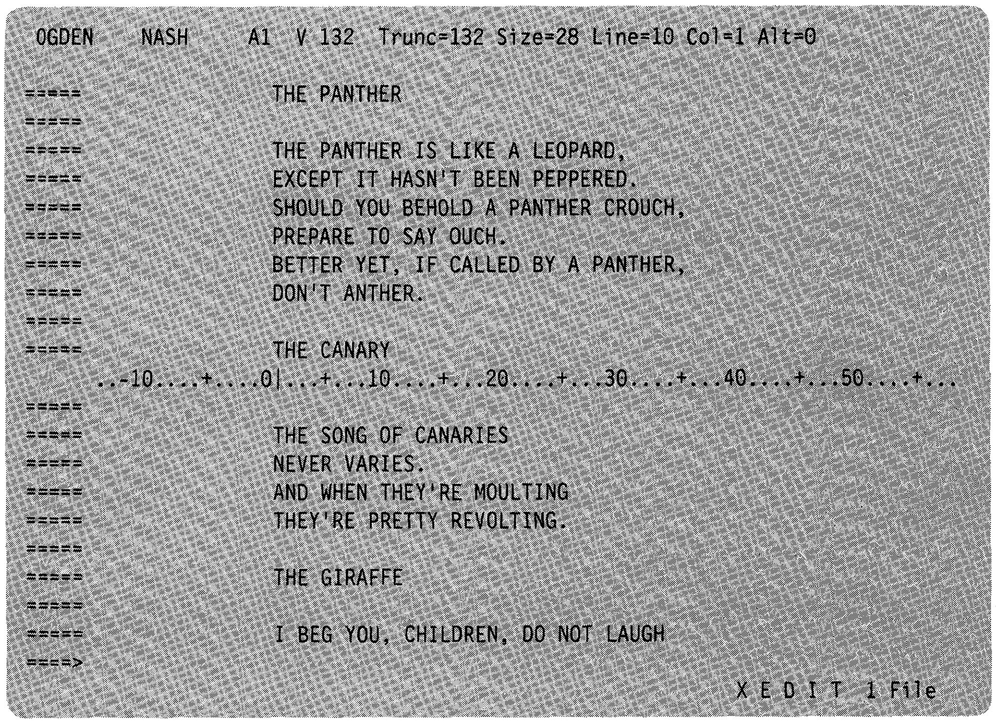
0	Normal
5	Invalid operand or number, or total verify width exceeds screen size
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 8 is a before-and-after example of the LEFT subcommand.



```
 OGDEN    NASH     A1  V 132  Trunc=132 Size=28 Line=10 Col=1 Alt=0

===== THE PANTHER
=====
===== THE PANTHER IS LIKE A LEOPARD,
===== EXCEPT IT HASN'T BEEN PEPPERED.
===== SHOULD YOU BEHOLD A PANTHER CROUCH,
===== PREPARE TO SAY OUCH.
===== BETTER YET, IF CALLED BY A PANTHER,
===== DON'T ANTHER.
=====
===== THE CANARY
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
===== THE SONG OF CANARIES
===== NEVER VARIES.
===== AND WHEN THEY'RE MOULTING
===== THEY'RE PRETTY REVOLTING.
=====
===== THE GIRAFFE
=====
===== I BEG YOU, CHILDREN, DO NOT LAUGH
====> left 15
                                                               X E D I T  1 File
```



```
 OGDEN    NASH     A1  V 132  Trunc=132 Size=28 Line=10 Col=1 Alt=0

=====                THE PANTHER
=====
=====                THE PANTHER IS LIKE A LEOPARD,
=====                EXCEPT IT HASN'T BEEN PEPPERED.
=====                SHOULD YOU BEHOLD A PANTHER CROUCH,
=====                PREPARE TO SAY OUCH.
=====                BETTER YET, IF CALLED BY A PANTHER,
=====                DON'T ANTHER.
=====
=====                THE CANARY
      ..-10....+....0|...+...10....+...20....+...30....+...40....+...50....+...
=====
=====                THE SONG OF CANARIES
=====                NEVER VARIES.
=====                AND WHEN THEY'RE MOULTING
=====                THEY'RE PRETTY REVOLTING.
=====
=====                THE GIRAFFE
=====
=====                I BEG YOU, CHILDREN, DO NOT LAUGH
====>
                                                               X E D I T  1 File
```

Figure 8. The LEFT Subcommand -- Before and After

LOAD

Use the LOAD subcommand to read a copy of the file being edited into virtual storage. *The LOAD subcommand can be issued only from the XEDIT profile.* Its purpose is to allow the profile (macro) to prompt the user for editing options or to assign default values for editing variables. The LOAD subcommand has the same format and editing options as the XEDIT command; however, the options specified in the XEDIT command override those specified in the LOAD subcommand.

Format

```
LOAD      [fn [ft [fm]]]      [(options...[ )] ]

          Options:
            [Width nn]  [NOSCreen]  [PROFile macroname]
            [NOPROFil]  [NOCLear]  [NOMsg]
            [MEMber membername]  [WINdow wname]

            [LOCk   ]
            [NOLOCk ]

          Options Valid Only in Update Mode:

            [Update  ]   [Seq8  ]   [Ctl fn1]
            [NOUpdate]   [NOSeq8]   [NOCtl  ]

            [Merge]   [UNtil filetype]   [Incr nn]
            [SIDcode string ]
```

For a description of the editing options, refer to the XEDIT command description. Remember that the options specified with the XEDIT command (or subcommand) override those specified with the LOAD subcommand.

Usage Notes

1. WINdow wname is the name of the virtual screen and window that XEDIT will use to display the file or files being edited. For more information on virtual screens and windows, see Appendix G, "XEDIT Virtual Screens and Windows" on page 441.
2. The WIDTH option specifies the amount of virtual storage to be used for one file line.

   If, upon editing an existing file, the value of WIDTH specified in the LOAD subcommand is too small to contain a file line, it is overridden by:

   a. The WIDTH value, if specified, in the XEDIT command.

   b. The logical record length of the file being read.

   Therefore, the value of WIDTH specified in the LOAD subcommand cannot cause unwanted truncation.

For newly created files, WIDTH will be the value specified on the LOAD subcommand unless it is overridden by the value specified on the XEDIT command.

If WIDTH is not specified in either the LOAD subcommand or the XEDIT command, its value is the greater of:

a. The logical record length (LRECL) of the file, or
b. The default logical record length associated with the file type. See Appendix A, "File Type Defaults" on page 403.

Note that WIDTH is the only option that has a different meaning in the LOAD subcommand and XEDIT command.

3. The following XEDIT variables are assigned default values when the LOAD subcommand is executed. These variables can be changed during editing by the SET subcommand:

```
LRECL
TRUNC
ZONE
VERIFY
```

`LRECL`
: is the logical record length that is used when the file is written to disk or directory. For files with a variable (V) record format, the LRECL is assigned the value specified in WIDTH. Thus, the longest record cannot exceed the value of WIDTH.

  For files with a fixed (F) record format, the LRECL assigned is one of the following:

  - If the file existed and has been read from a disk or a directory, the LRECL assigned is the logical record length of the file.
  - If the file is new, LRECL is either WIDTH or the default logical record length assigned to the file type, whichever is smaller.

`TRUNC`
: is assigned the value of LRECL, except for fixed (F) format files, which have a default value associated with the file type.

`ZONE`
: is initially set to 1 (zone1) and TRUNC (zone2).

`VERIFY`
: is assigned the value of TRUNC unless otherwise specified (See Appendix A, "File Type Defaults" on page 403).

For a list of file type defaults, see Appendix A, "File Type Defaults" on page 403.

4. Within the profile macro, the LOAD subcommand must be the first XEDIT subcommand. If it is not, a LOAD subcommand is automatically issued by the editor; its operands are the same as those issued in the XEDIT command. (EXEC 2 statements, REXX instructions, and CMS commands can be issued before the LOAD. When a CMS command is issued before the LOAD, the command must be specifically addressed to CMS. In REXX, use ADDRESS CMS and in EXEC 2 use &COMMAND.)

A synonym cannot be used to specify the LOAD subcommand. Multiple subcommands may not be specified with the linend character following the LOAD subcommand.

5. The profile macro can be used to prompt the user for XEDIT command options or to assign values to editing variables before issuing the LOAD subcommand. For example, a SCRIPT user might program his profile to use a LOAD subcommand that does defaulting of file type. For example:

   ```
   parse arg fn ft '(' options
   if ft = '' then ft = 'SCRIPT'
   LOAD fn ft '(' options
   ```

6. The options specified in the LOAD subcommand have a *lower* priority than those in the XEDIT command. For example, UPDATE specified in the LOAD subcommand would be overridden by NOUPDATE specified in the XEDIT command.

   Thus, with the proper profile, all options in the XEDIT command (or subcommand) can be made optional.

   As a general rule, options in the LOAD subcommand indicate general user preferences that can be overridden by options specified in the XEDIT command itself.

7. If the LOAD is unsuccessful, a non-zero return code is generated. All subsequent subcommands in the profile are rejected with a unique "6" return code, and the editor automatically issues a QUIT subcommand.

Responses

When editing a file that resides in an SFS directory, if you have only read authority to the file and you do not specify the NOLOCK option, you will receive the message:

```
1299W Warning: not authorized to lock fn ft fm
```

The editing session will continue but the file will not be locked.

The following messages are displayed only if you are using XEDIT in update mode:

```
178I Updating fn ft fm
     Applying fn ft fm
           .
           .
           .
180W  Missing PTF file fn ft fm
```

Messages

002E	File *fn ft fm* not found [RC=28]
003E	Invalid option: *option* [RC=24]
024E	File XEDTEMP CMSUT1 A1 already exists [RC=28]
029E	Invalid parameter *parameter* in the option *option* field [RC=24]
048E	Invalid mode *mode* [RC=24]
054E	Incomplete fileid specified [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
065E	*option* option specified twice [RC=24]
066E	*option1* and *option2* are conflicting options [RC=24]
069E	Filemode *mode* not accessed [RC=36]
070E	Invalid parameter *parameter* [RC=24]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, or 100]
109S	Virtual storage capacity exceeded [RC=104]
132S	File *fn ft fm* too large [RC=88]
229E	Unsupported OS dataset, error *nn* [RC=80, 81, 82, or 83]
500E	Unable to unpack file *fn ft fm* [RC=88]
508E	LOAD must be the first subcommand in the profile [RC=3]
554E	Not enough virtual storage available [RC=104]
555E	File *fn ft fm* already in storage [RC=4]
571I	Creating new file:
622E	Insufficient free storage for {MSGLINE|PFkey/PAkey|synonyms}
915E	Maximum number of windows already defined [RC=13]
927E	The virtual screen must contain at least 5 lines and 20 columns [RC=24]
928E	Command is not valid for virtual screen *CMS* [RC=12]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1214W	File *fn ft fm* already locked SHARE
1215E	File *fn ft fm* is locked {SHARE|UPDATE|EXCLUSIVE} by another user [RC=70]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 70, 76, 99, or 100]
1299W	Warning: Not authorized to lock file *fn ft fm*
1300E	Error *nn* {locking|unlocking} file *fn ft* {*fm*|*dirname*} [RC=55, 70, 76, 99, or 100]

Messages with Member Option

007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
033E	File *fn ft fm* is not a library [RC=32]
039E	No entries in library *fn ft fm* [RC=32]
167S	Previous MACLIB function not finished [RC=88]
622E	Insufficient free storage for reading map [RC=104]

Messages with Update Options

007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
007E	File *fn ft fm* does not have a logical record length greater than or equal to 80 [RC=32]
007E	File *fn ft fm* does not have the same format and record length as *fn ft fm* [RC=32]
007E	File *fn ft fm* is not fixed record format [RC=32]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 32, or 55]
174W	Sequence error introduced in output file: *seqno1* to *seqno2* [RC=32]
178I	Applying *fn ft fm*
179E	Missing or invalid MACS card in control file *fn ft fm*
180W	Missing PTF file *fn ft fm*
183E	Invalid {CONTROL|AUX} file control card [RC=32]
184W	./ S not first card in update file--ignored [RC=32]
185W	Non numeric character in sequence field *seqno* [RC=32]
186W	Sequence number [*seqno1*] not found [RC=32]
207W	Invalid update file control card [RC=32]
210W	Input file sequence error: *seqno1* to *seqno2* [RC=32]
570W	Update *ft* specified in the UNTIL option field not found
597E	Unable to merge updates containing ./S cards [RC=32]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 70, 76, 99, or 100]

Return Codes

0	Normal
3	LOAD has already been issued
4	File is already in storage
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Command is not valid for virtual screen
13	Maximum number of windows already defined
20	Invalid character in file name or file type
24	Invalid parameters or options
28	Source file not found (UPDATE MODE), or library not found (MEMBER option), or specified PROFILE macro does not exist, or file XEDTEMP CMSUT1 already exists
31	A rollback occurred
32	Error during updating process, or file is not a library, or library has no entries, or file is not fixed, 80 char. records
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
88	File is too large and does not fit into storage, or previous Maclib function was not finished
99	A required system resource is not available

100	Error reading the file into storage
104	Insufficient storage available

LOCATE

Use the LOCATE subcommand to scan the file for a specified target, which (if found) becomes the new current line. The search starts with the line following the current line. Optionally, an XEDIT subcommand may be specified; it is executed starting at the specified target.

Note: To display **all** occurrences of a given string, use the ALL macro.

Format

[**Locate**]	*target* [*subcommand*]

Operands

Locate
is the subcommand name but is optional. The target operand itself implies the LOCATE subcommand.

target
defines the line that is to become the new current line. It can be specified as an absolute line number, a relative displacement from the current line, a line name, a simple string expression, or a complex string expression. A complete description of targets follows.

subcommand
is any XEDIT subcommand, which is executed starting at the specified target.

Usage Notes – Targets

The ability to locate a line via a target is one of the editor's most versatile functions. A target is used not only as the operand of the LOCATE subcommand but also as an operand in many other XEDIT subcommands.

A target is a way to define a line to be searched for within the current top and bottom of the file (or top and bottom of the range – see SET RANGE) and between the beginning and end of each line (or between the left and right zones – see SET ZONE).

A target can be as simple or as complex as the user desires. It can be expressed in the following ways:

1. An *absolute line number* is a colon (:) followed by a file line number. For example:

   ```
   :8
   ```

   makes file line number 8 the new current line.

2. A *relative displacement* from the current line is an integer and may be preceded by a plus (+) or minus (–) sign, which indicates a forward (+) or backward (–) displacement from the current line. If the sign is omitted, a plus (+) is assumed.

   A relative displacement may also be specified as an asterisk (*), which means the Top of File (–*) or the End of File (+* or *) line. When an asterisk is specified as the target operand of a subcommand, the subcommand executes to the end (or top) of the file. For example, DELETE * deletes lines from the current line to the end of the file. Examples of relative displacement follow:

`+3`
: The target is three logical lines down (toward the end of the file) from the current line.

`-5`
: The target is five logical lines up (toward the top of the file) from the current line.

`+*`
: The target is the null End of File (or End of Range) line.

`-*`
: The target is the null Top of File (or Top of Range) line.

```
:5 copy +3 :25
```

requests that lines 5, 6, and 7 be copied after line number 25.

In this example, three targets are specified. The first (:5) is a LOCATE, even though the subcommand name is not specified.

3. A *line name* is one to eight characters preceded by a period (.), which has been previously defined by a SET POINT subcommand or a .xxxx prefix subcommand (which limits the name to four characters). For example:

   ```
   .PART2
   ```

   locates the file line whose name is PART2 and makes it the current line.

   The SET CASE (UPPERCASE or MIXED) option determines whether the line name is translated to uppercase. Additionally, in locating a line name target, uppercase and lowercase characters are significant regardless of the SET CASE option RESPECT or IGNORE.

4. A *string expression* defines a group of characters to be located. The characters in the string must be delimited by any character that does not appear in the string itself. However, if the XEDIT special characters (+ − .) are used to delimit a string target, the search direction (+ or −) must be stated explicitly. When a string target is entered by itself (without the optional subcommand name LOCATE), the delimiter must be a diagonal (/). In the following examples, a diagonal (/) is used.

   If SET HEX ON has been issued, a string may be specified in hexadecimal notation, and the editor searches for its EBCDIC equivalent. For example, if a string is specified as /X'C3D4E2'/, the editor searches for the string "CMS".

   The general format for a string expression is:

   ```
   [+|-][¬]/string1[/&[¬]/string2/[|[¬]/string3]]...
    a  b   c       d               e
   ```

   a. The search direction is toward the end of the file (+) or toward the top of the file (−). If the sign is omitted, a plus (+) is assumed.

   b. "NOT" symbol (Locate something that is not the specified string.)

c. Character (or hexadecimal) string. The trailing delimiter may be necessary in certain circumstances. For example, if string1 has trailing blanks, the trailing delimiter should be used to indicate where the string ends.

d. "AND" symbol (ampersand) (Locate the line containing string1 and string2.)

e. "OR" symbol (vertical bar) (Locate any of the strings, separated by OR symbols, starting with the first string specified.)

For example:

`/horse/`
: searches downward in the file, beginning after the current line, for the first line that contains "horse" and makes it the current line.

`¬/house/`
: searches downward in the file for the first line that does *not* contain "house" and makes it the current line.

`/horse/ & /house/ | /hay/`
: searches downward in the file for a line that contains either both "horse" *and* "house" *or* a line that contains "hay," whichever comes first.

  Targets that are "anded" together can overlap. For example, "L/This/&/history/" could find the string "Thistory" as well as the string "This history".

`/horse/|¬/house/`
: searches downward in the file for the first line that contains "horse" *or* does not contain "house."

`-/X'C1'/|/X'C2'/`
: searches upward for the first line containing *either or both* of the strings specified here in hexadecimal.

  If SET HEX ON is in effect, the editor locates a line containing "A" or "B." If SET HEX OFF is in effect, the editor locates a line containing "X'C1'" or "X'C2'."

`//`
: advances the line pointer by one line.

5. A *complex string expression* has the same format as a simple string expression. A string can be expressed as a "complex string" by associating it with one or more of the following SET subcommand options:

`SET ARBCHAR`
: allows you to specify only the beginning and end of a string, using an arbitrary character to represent all characters in the middle.

`SET CASE`
: allows you to specify whether or not the difference between uppercase and lowercase is to be significant in locating a string target.

`SET ETARBCH`
: See Appendix F.

`SET SPAN`
: allows you to specify if a string target must be included in one file line or if it may span a specified number of lines.

`SET VARBLANK`
: allows you to control whether or not the number of blank characters between two words is significant in a target search.

For example:

LOCATE and SET ARBCHAR

```
set arbchar on .
/air.plane/
```

would locate in the text either one of the following:

```
"the airplane was landing"
"cold air surrounded the plane"
```

LOCATE and SET CASE

```
set case m ignore
/computer/
```

would locate in the text any of the following:

```
"computer"
"Computer"
"comPUTer"
```

LOCATE and SET SPAN

If SET SPAN OFF is in effect, a string must be contained within one file line in order to match a target.

If SET SPAN ON is in effect, a string may start on one line and continue on n lines (as specified in the SET SPAN subcommand) and still match a target.

LOCATE and SET VARBLANK

```
set varblank on
/the  house/
```

will locate in the text either of the following:

```
"the house"
"the              house"
```

Note: Target as discussed here is actually a *line target* and is not to be confused with a *column-target,* which is used as the operand of only the CLOCATE and CDELETE subcommands.

Usage Notes for LOCATE Targets

1. LOCATE targets and SET STAY: If SET STAY OFF is in effect and the target is not located, the new current line is the bottom (or top) of the file (or range).

   If SET STAY ON is in effect and the target is not located, the line pointer remains unchanged.

2. LOCATE targets and SET WRAP: If SET WRAP OFF is in effect, the search for a string expression target stops with the end of file or range (or top of file or range).

   If SET WRAP ON is in effect, the editor wraps around the file and continues the search for a string expression up to and including the line preceding the current line. If a range has been defined, the editor wraps around the bottom of the range and continues at the top of the range (or if the search is in the other direction, the editor wraps around the top and continues at the bottom).

When a wrap occurs under these conditions, the following warning message is displayed:

```
592W Wrapped ....
```

3. The LOCATE subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Notes for Macro Writers

In a macro, an implicit backward locate (for example, −3) is interpreted by EXEC 2 as a label. To avoid this problem, use the LOCATE subcommand explicitly (for example, LOCATE −3), or use the COMMAND subcommand (COMMAND −3).

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
592W	Wrapped ....

Return Codes

0	Normal
1	TOF or EOF reached
2	No target line was found
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
nn	Return code from subcommand following LOCATE

LOWERCAS

Use the LOWERCAS subcommand to change all uppercase letters to lowercase in one or more lines of the file, starting with the current line.

Format

LOWercas	[*target* |**1**]

Operand

target
defines the number of lines to be translated. Translation starts with the current line and continues up to, but does not include, the target line. If you specify an asterisk (*), the rest of the file is translated. If you omit target, only the current line is translated.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Note

The LOWERCAS subcommand does not alter the setting of the SET CASE subcommand.

Responses

1. All uppercase letters within the current zones are changed to lowercase.
2. If you specify that a LOWERCAS is to occur on multiple lines and the LOWERCAS occurs, the current line pointer will be:
   1) Unchanged, if SET STAY ON has been issued
   2) Moved to the last line translated, if SET STAY OFF is in effect (the default).

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
4	No line(s) changed
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

```
===== Tortoises of the Galapagos Islands can live to be 100 years old.
```

`low` (translate the current line to lowercase)

```
===== tortoises of the galapagos islands can live to be 100 years old.
```

LPREFIX

Use the LPREFIX subcommand to simulate writing in the prefix area of the current line. LPREFIX can be used on typewriter terminals to utilize some of the features of prefix subcommands and macros, as well as on display terminals, irrespective of the appearance or position of actual prefix areas on the screen (see the SET PREFIX subcommand).

Format

LPrefix	[*text*]

Operand

text
specifies up to five characters. All prefix subcommands and macros are put in a "pending list" before they are called. If no text is specified for LPREFIX, the pending list is executed immediately. If text is specified, the text is placed in the pending list for the current line and the pending list is then executed. For more information about the pending list, refer to the SET PENDING subcommand.

Usage Notes

1. Note that the action taken after LPREFIX is identical to the action taken when you enter the same "text" subcommand directly in the prefix area of the current line and press ENTER.
2. "LPREFIX text" is equivalent to issuing "SET PENDING ON string" and pressing ENTER again.
3. LPREFIX cannot be issued from a prefix macro.
4. The prefix subcommands and macros that are useful on a typewriter terminal via LPREFIX are:

```
D,DD
",""
C,CC
M,MM
X,XX
<,<<
>,>>
.XXXX
P
F
```

Messages

509E	*subcommand* subcommand not valid from a prefix macro [RC=4]
520E	Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
4	Invalid when issued from a prefix macro
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

Nefarious Nelly put her grocery list online. Every day she added more items. At the end of the week, she wanted to organize the list in the following order: meat, fruit, vegetables. Working at a typewriter terminal, she used the LPREFIX subcommand to move the grocery items appropriately. The following is a listing of her file's contents:

```
TOF:
Toad Toes
Newt Eyes
Salmon Scales
Root of Hemlock
Baneberries
Poison Plums
Paltry Peaches
EOF:
```

Nelly moves to the top of the file by typing "top" and then moves down to the line "BANEBERRIES" by typing "n5".

```
top
```

```
TOF:
```

```
n5
```

```
Baneberries
```

To move the last three lines of the file (the fruit) so that they follow Salmon Scales (the last line of the meat group), Nelly used the LPREFIX subcommand as shown in the following sequence:

```
lprefix mm
```

```
n2
```

```
Paltry Peaches
```

```
lprefix mm

top
```

```
TOF:
```

```
n3
```

```
Salmon Scales
```

```
lprefix f
```

The resulting file:

```
top
```

```
TOF:
```

```
t*
```

```
TOF:
Toad Toes
Newt Eyes
Salmon Scales
Baneberries
Poison Plums
Paltry Peaches
Root of Hemlock
EOF:
```

MACRO

Use the MACRO subcommand to cause the specified operand to be executed as a macro.

Format

MACRO	[*macroline*]

Operand

macroline
is an XEDIT macro name and its arguments (if any).

The first word in the macroline is assumed to be an XEDIT macro name. It starts with the first non-blank character following the subcommand name MACRO, and it ends with the first character followed by a blank. The name can be from one to eight characters long; it is truncated to eight characters, if necessary.

Usage Notes

1. The MACRO subcommand causes the editor to execute the specified macro without first checking to see if a subcommand of the same name or a synonym exists.
2. The MACRO subcommand must be used when the macro name contains non-alphabetic characters. It does not follow the usual parsing rule of separating alphabetic characters from immediately following non-alphabetic characters. For example, N2 usually means NEXT 2. MACRO N2 means "execute the macro named N2."

Responses

The response, if any, from the executed macro is displayed.

Message

542E No such subcommand: *name* [RC=-1]

Return Codes

−1	No such subcommand
nn	Return code of the macro specified as operand
0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

1. macro N ABC
   invokes the macro file N XEDIT with the argument ABC.
2. macro N2 ABC
   invokes the macro file N2 XEDIT with the argument ABC.

MERGE

Use the MERGE subcommand to combine two sets of lines. The first set of lines is deleted and the second set is modified in place.

Format

MErge	*target* **1** *target* **2** [*col*]

Operands

target1
defines the *number* of lines to be merged, starting with the current line up to, but not including, target1. Target1 defines the first group of lines that you wish to merge with a second group of lines.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication, *VM/SP System Product Editor User's Guide.*

target2
defines the beginning of the second group of lines that is to be merged with the first.

col
specifies the column number to which column 1 of the first group of lines being merged is to be shifted. The first group is shifted to the right of the second group and then overlays the second group. This allows the merger of two columns of data, side by side. The default "col" is 1.

Usage Notes

1. MERGE shifts the first group of lines and then overlays the second group of lines. It is similar in function to the OVERLAY subcommand, except that MERGE does not give special treatment to underscore characters.
2. Following the MERGE with the second set of lines, the first set of lines is deleted. The lines can be recovered by using the RECOVER subcommand.
3. The first group of lines cannot overlap the second group of lines.
4. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.

5. When combining two lines, the merge handles blanks as follows:

   ```
   First line    Second line    Result line
     blank         blank          blank
       x           blank            x
     blank           y              y
       x             y              x
   ```

   where 'First line' designates any of the lines in the group starting at the current line, and 'Second line' designates any of the lines in the group starting at the second target parameter. The blank columns of the line to be modified are replaced with the corresponding columns from the line in the first merge group; however, blanks in the first merge group do not replace character data in the line to be modified.

Response

The last line merged becomes the new current line.

Messages

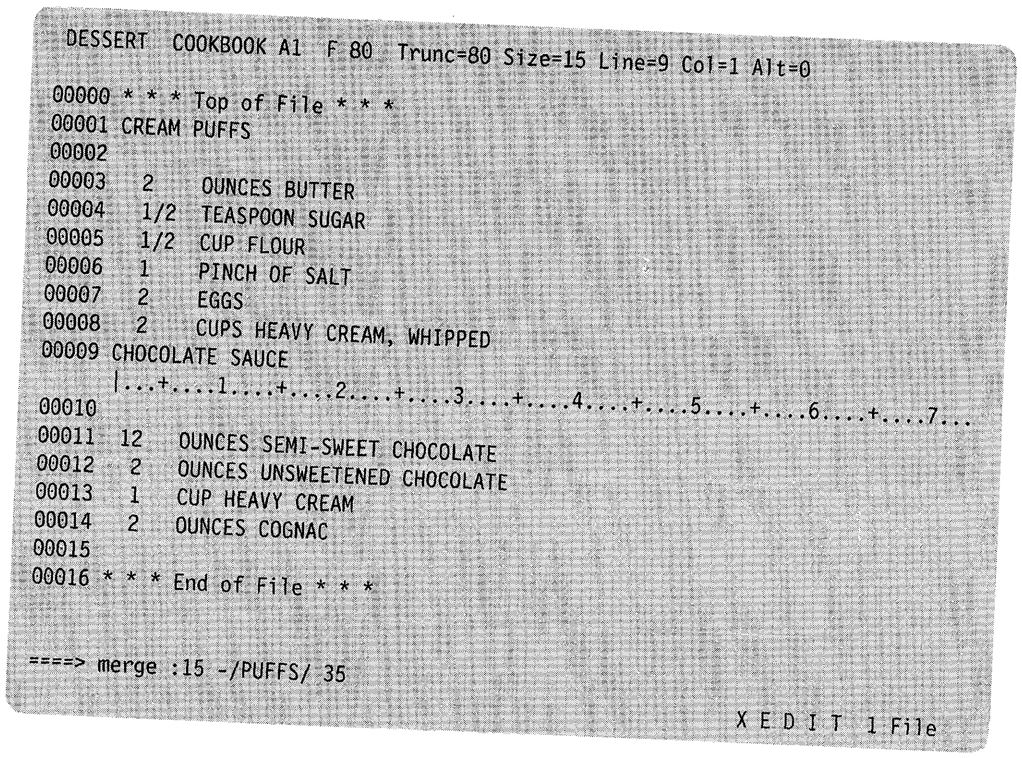
498E	Not executed--the two areas to merge overlap each other [RC=1]
520E	Invalid operand: *operand* [RC=5]
527E	Invalid column number [RC=1]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
581E	Subcommand is not valid in extended mode [RC=3]
593E	{No|*nn*} lines merged, *nn* line(s) {truncated|spilled} [RC=3]

Return Codes

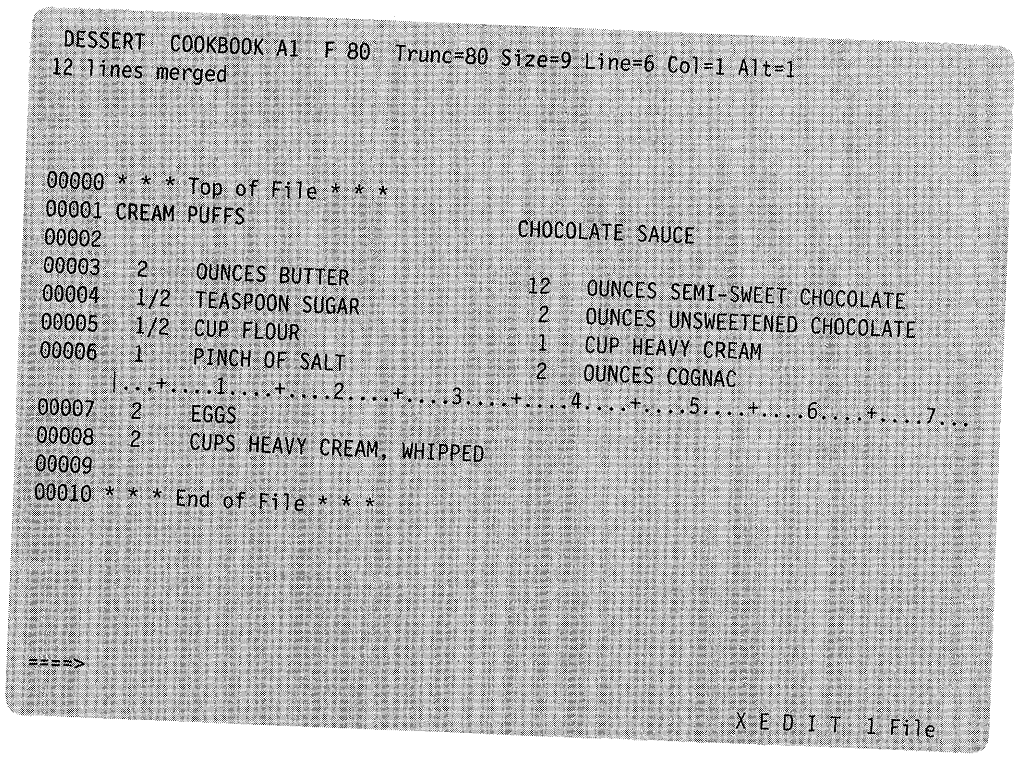
0	Normal
1	Overlapping groups of lines; invalid column number
2	Target line not found
3	Truncated or spilled, or subcommand is not valid in extended mode
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 9 is a before-and-after example of the MERGE subcommand.



```
 DESSERT  COOKBOOK A1  F 80  Trunc=80 Size=15 Line=9 Col=1 Alt=0

00000 * * * Top of File * * *
00001 CREAM PUFFS
00002
00003   2    OUNCES BUTTER
00004   1/2  TEASPOON SUGAR
00005   1/2  CUP FLOUR
00006   1    PINCH OF SALT
00007   2    EGGS
00008   2    CUPS HEAVY CREAM, WHIPPED
00009 CHOCOLATE SAUCE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00010
00011  12   OUNCES SEMI-SWEET CHOCOLATE
00012   2   OUNCES UNSWEETENED CHOCOLATE
00013   1   CUP HEAVY CREAM
00014   2   OUNCES COGNAC
00015
00016 * * * End of File * * *


====> merge :15 -/PUFFS/ 35
                                                              X E D I T  1 File
```



```
 DESSERT  COOKBOOK A1  F 80  Trunc=80 Size=9 Line=6 Col=1 Alt=1
12 lines merged



00000 * * * Top of File * * *
00001 CREAM PUFFS
00002                                     CHOCOLATE SAUCE
00003   2    OUNCES BUTTER
00004   1/2  TEASPOON SUGAR                12   OUNCES SEMI-SWEET CHOCOLATE
00005   1/2  CUP FLOUR                      2   OUNCES UNSWEETENED CHOCOLATE
00006   1    PINCH OF SALT                  1   CUP HEAVY CREAM
                                            2   OUNCES COGNAC
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00007   2    EGGS
00008   2    CUPS HEAVY CREAM, WHIPPED
00009
00010 * * * End of File * * *






====>
                                                              X E D I T  1 File
```

Figure 9. The MERGE Subcommand – Before and After

MODIFY (Macro)

Use the MODIFY macro to display a subcommand and its current operand values in the command line, so that a new operand value can be typed over the current one and the subcommand immediately reentered.

Format

MODify	*keyword*

Operand

keyword
is one of the following keyword operands valid with the SET, QUERY, EXTRACT, or TRANSFER subcommands:

```
 ALT              HEX                     SCReen
 APL              IMage                   SELect
 ARBchar          IMPcmscp                SERial
 AUtosave         LASTLorc                SHADow
 BRKkey           LINENd                  SIDcode
 CASE             LRecl                   SPAN
 CMDline          MACRO                   SPILL
*COLOR field      MASK                    STAY
 COLPtr           MSGLine                 STReam
 COLumn           MSGMode                 SYNonym
 CTLchar [char]   NONDisp                 TABLine
 CURLine          NULls                   TABS
 DISPlay          NUMber                  TERMinal
 ENTer            PAn                     TEXT
 ESCape           PACK                    TOFEOF
 ETARBCH          PFn                     TRunc
 ETMODE           PREfix [Synonym name]   VARblank
 FILler           RANge                   Verify
 FMode            RECFm                   VERShift
 FName            REMOte                  WRap
 FType            SCALe                   Zone
 FULLread         SCOPE
```

* Refer to the SET COLOR subcommand in this publication for a list of the fields.

Usage Notes

1. All of the above keywords cause the corresponding SET subcommand and its current operand value to be displayed, except for the following:

   `modify column`
   displays CLOCATE :nn, where: nn is the current column.

   `modify mask`
   displays the current mask, so that you can type over it to modify it.

   `modify synonym`
   displays SET SYNONYM ON or SET SYNONYM OFF.

2. MODIFY CTLCHAR can be specified only when no control characters are defined. Otherwise, specify MODIFY CTLCHAR char.

Response

If the keyword specified is unknown or is an XEDIT variable that cannot be modified, an error message is displayed.

Messages

520E	Invalid operand: *operand* [RC=5]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
3	Subcommand valid only in display mode
5	Invalid or missing operand(s)
6	Subcommand rejected in the PROFILE due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

1. `mod zone`

   displays

   ```
   SET ZONE 5 25
   ```

   You can type over the "25" and change it to "50." Then press the ENTER key. The new zone settings will be 5 and 50.

2. `mod nulls`

   displays

   ```
   SET NULLS OFF
   ```

   Type "ON" – to set NULLS ON.

MOVE

Use the MOVE subcommand to move one or more lines, beginning with the current line, to a specified position in the file. The original lines are deleted.

Format

MOve	*target* **1** *target* **2**

Operands

target1
defines the number of lines to be moved. The lines are removed from the file starting with the current line up to, but not including, target1.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*

target2
defines the destination line. The data is moved *after* target2.

Responses

The last line moved becomes the new current line.

The editor displays the following message:

```
506I nn lines moved
```

Messages

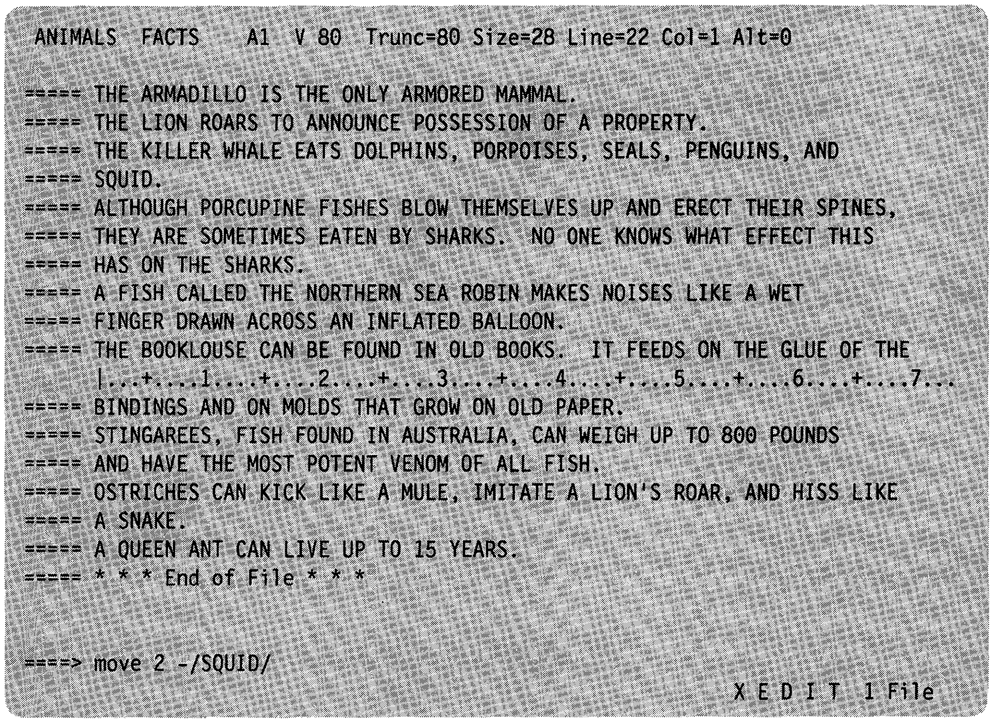
505E	Not executed--the target line (*nn*) is within the lines to move [RC = 1]
520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]
546E	Target not found [RC = 2]

Return Codes

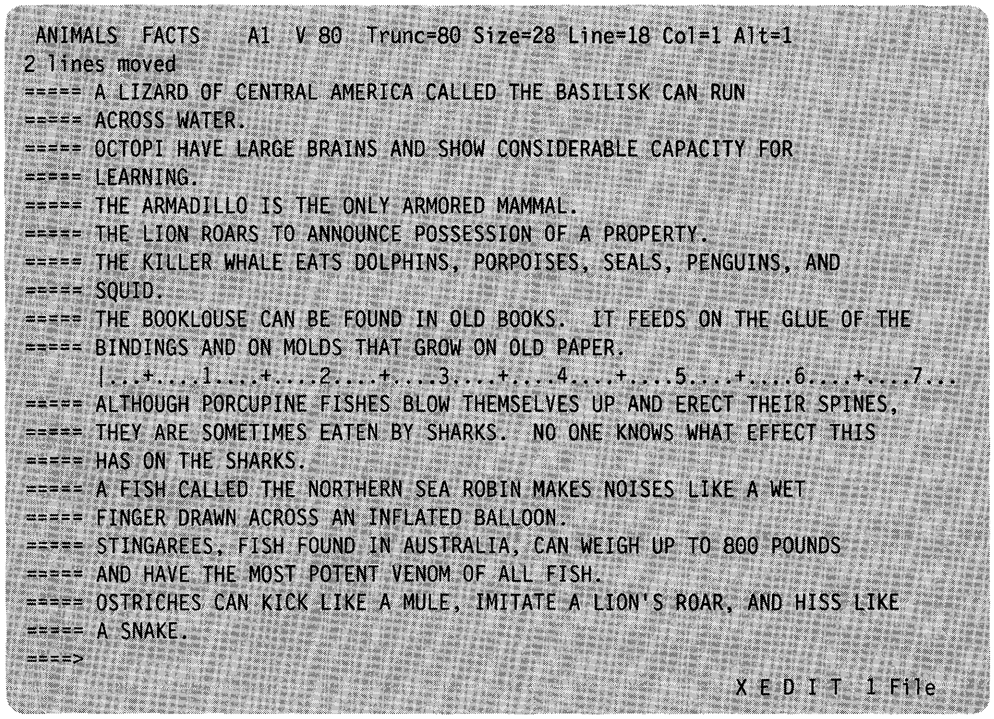
0	Normal
1	Target line within the lines to move
2	Target line not found
5	Invalid or missing operands
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 10 is a before-and-after example of the MOVE subcommand.



```
 ANIMALS  FACTS    A1  V 80  Trunc=80 Size=28 Line=22 Col=1 Alt=0

===== THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
===== HAS ON THE SHARKS.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
===== FINGER DRAWN ACROSS AN INFLATED BALLOON.
===== THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS.  IT FEEDS ON THE GLUE OF THE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
===== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
===== AND HAVE THE MOST POTENT VENOM OF ALL FISH.
===== OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISS LIKE
===== A SNAKE.
===== A QUEEN ANT CAN LIVE UP TO 15 YEARS.
===== * * * End of File * * *


====> move 2 -/SQUID/
                                                       X E D I T  1 File
```



```
 ANIMALS  FACTS    A1  V 80  Trunc=80 Size=28 Line=18 Col=1 Alt=1
2 lines moved
===== A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
===== THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== THE BOOKLOUSE CAN BE FOUND IN OLD BOOKS.  IT FEEDS ON THE GLUE OF THE
===== BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
===== HAS ON THE SHARKS.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
===== FINGER DRAWN ACROSS AN INFLATED BALLOON.
===== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
===== AND HAVE THE MOST POTENT VENOM OF ALL FISH.
===== OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISS LIKE
===== A SNAKE.
====>
                                                       X E D I T  1 File
```

Figure 10. The MOVE Subcommand – Before and After

MSG

Use the MSG subcommand to display a message in the message area of the logical screen.

Format

MSG	[*text*]

Operand

text
is the text of the message to be displayed. If omitted, a blank line will be displayed.

Usage Notes

1. The MSG subcommand does not sound the alarm. (Use the EMSG subcommand to sound the alarm.)
2. MSG can also be used to type a message on a typewriter terminal.

Notes for Macro Writers

1. Use the MSG subcommand within a macro to display a message at the terminal.
2. When multiple messages are issued and they do not fit on the message line(s) as defined by SET MSGLINE, they are passed to CMS. If full-screen CMS is OFF, the message replaces the file image on the screen and the terminal is placed in a "MORE..." (waiting) status. To get the file image back on the screen, press the CLEAR key.

   If full-screen CMS is ON, the message will appear in the CMSOUT window. To see all of the text in the virtual screen, you can use the CMS SCROLL command. The screen will be cleared automatically when you scroll to the bottom of the virtual screen. An alternative way to clear the screen is to issue the CMS DROP WINDOW command. If you have deleted the CMSOUT window, you won't see messages that are passed to CMS.

Responses

The message is displayed in the message area of the screen.

Return Codes

0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

NEXT

Use the NEXT subcommand to advance the line pointer a specified number of lines toward the end of the file. The line pointed to becomes the new current line. (The NEXT subcommand is equivalent to the DOWN subcommand.)

Format

Next	[*n* | * | 1]

Operand

n

is the number of lines to move the line pointer. If you specify an asterisk (*), the line pointer moves to the "End of File" line. If you omit n, the pointer moves down only one line.

Usage Note

The `NEXT` *n* subcommand is equivalent to a plus (+) target definition.

For example:

```
next 3
```

is equivalent to

```
+3
```

Response

The line pointed to becomes the new current line.

Messages

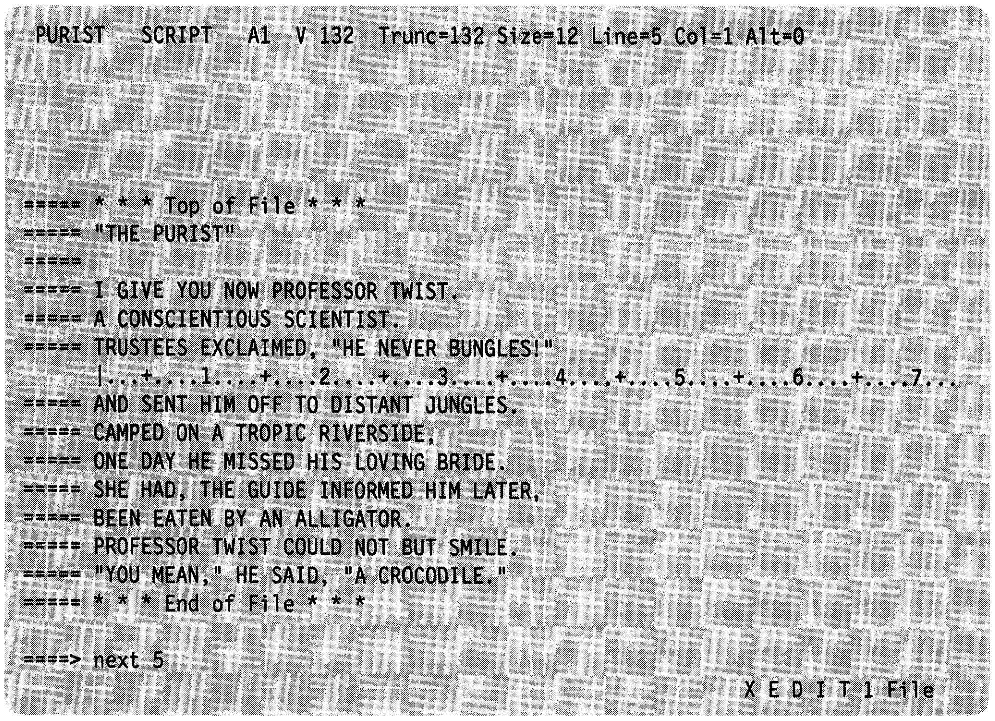
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]

Return Codes

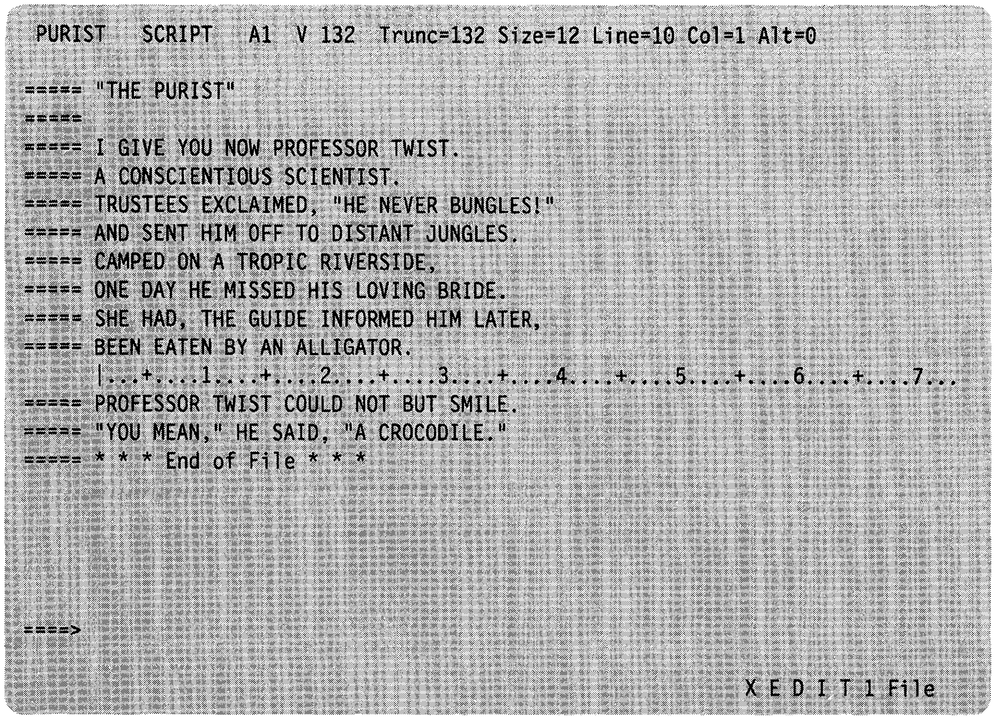
0	Normal
1	End of file reached and displayed
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 11 is the before-and-after example of the NEXT subcommand.



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=5 Col=1 Alt=0




===== * * * Top of File * * *
===== "THE PURIST"
=====
===== I GIVE YOU NOW PROFESSOR TWIST.
===== A CONSCIENTIOUS SCIENTIST.
===== TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== AND SENT HIM OFF TO DISTANT JUNGLES.
===== CAMPED ON A TROPIC RIVERSIDE,
===== ONE DAY HE MISSED HIS LOVING BRIDE.
===== SHE HAD, THE GUIDE INFORMED HIM LATER,
===== BEEN EATEN BY AN ALLIGATOR.
===== PROFESSOR TWIST COULD NOT BUT SMILE.
===== "YOU MEAN," HE SAID, "A CROCODILE."
===== * * * End of File * * *

====> next 5
                                                      X E D I T 1 File
```



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=10 Col=1 Alt=0

===== "THE PURIST"
=====
===== I GIVE YOU NOW PROFESSOR TWIST.
===== A CONSCIENTIOUS SCIENTIST.
===== TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
===== AND SENT HIM OFF TO DISTANT JUNGLES.
===== CAMPED ON A TROPIC RIVERSIDE,
===== ONE DAY HE MISSED HIS LOVING BRIDE.
===== SHE HAD, THE GUIDE INFORMED HIM LATER,
===== BEEN EATEN BY AN ALLIGATOR.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== PROFESSOR TWIST COULD NOT BUT SMILE.
===== "YOU MEAN," HE SAID, "A CROCODILE."
===== * * * End of File * * *






====>

                                                      X E D I T 1 File
```

Figure 11. The NEXT Subcommand – Before and After

NFIND

Use the NFIND subcommand to search forward in the file for the first line that does *not* start with the text specified in the operand. The search starts with the line following the current line.

Format

NFind	*text*

Operand

text
is any text that you do *not* want to find, beginning in column one of the next file line. Only the non-blank characters in the operand are checked against the file.

Usage Notes

1. Only one blank can be used as a delimiter following the NFIND subcommand.
2. If a SET IMAGE ON subcommand is in effect, tabs in the operand are changed to blanks (or filler characters) before the search is made. In addition, the search begins in the first tab column.
3. Use an underscore character (_) in the operand to designate that a blank character should appear in the line.
4. See also FIND, FINDUP, NFINDUP.
5. If the subcommand SET WRAP OFF has been issued, the search will continue until the end of file.

   If the subcommand SET WRAP ON has been issued, the search will continue through the entire file and stop when the line where it started is reached again. When a WRAP occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```

6. The NFIND subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Response

The first line that does *not* match the operand becomes the new current line.

Messages

545E	Missing operand(s) [RC=5]
586E	Not found [RC=2]
592W	Wrapped ....

Return Codes

0	Normal
2	No target line was found
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

NFINDUP

Use the NFINDUP subcommand to search *backward* in the file for the first line that does *not* start with the text specified in the operand. The search starts with the line preceding the current line.

Format

NFINDUp **NFUp**	*text*

Operand

text
is any text that you do *not* want to find, beginning in column one of the file line preceding the current line. Only the non-blank characters in the operand are checked against the file.

Usage Notes

1. Only one blank can be used as a delimiter following the NFINDUP subcommand.
2. If a SET IMAGE ON subcommand is in effect, tabs in the operand are changed to blanks (or filler characters) before the search is made. In addition, the search begins in the first tab column.
3. Use an underscore character (_) in the operand to designate that a blank character should appear in the line.
4. See also FIND, FINDUP, NFIND.
5. If the subcommand SET WRAP OFF has been issued, the search will continue until the top of file.

   If the subcommand SET WRAP ON has been issued, the search will continue through the entire file and stop when the line where it started is reached again. When a WRAP occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```

6. The NFINDUP subcommand updates the LASTLORC buffer. Refer to the SET LASTLORC subcommand in this publication.

Response

The first line that does *not* match the operand becomes the new current line.

Messages

545E	Missing operand(s) [RC = 5]
586E	Not found [RC = 2]
592W	Wrapped ....

Return Codes

0	Normal
2	No target line was found
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

OVERLAY

Use the OVERLAY subcommand to replace selectively one or more characters in the current line with the corresponding nonblank characters in the line being keyed in. If SET IMAGE ON is in effect, replacement starts at the first tab column of the current line.

Format

Overlay	*text*

Operand

text
specifies an input line that replaces corresponding character positions in the current line.

Usage Notes

1. Blank characters in the input line indicate that the corresponding characters in the current line are *not* to be overlaid.
2. Use an underscore character (_) in the input line to place a blank in the corresponding character position in the current line.
3. Use the CREPLACE subcommand instead of the OVERLAY subcommand when you want a one-for-one replacement of blanks and underscore characters.
4. At least one blank must follow the OVERLAY subcommand; the operand starts after the first blank that follows the subcommand name (or its abbreviation).
5. If SET IMAGE ON is in effect, tabs in the text operand are expanded to blank (or filler) characters. These blanks will also leave the corresponding characters in the current line unchanged.

   For example, the following subcommand adds a comment to an assembler language statement (file type ASSEMBLE) whose settings are defined by SET TABS 1 10 16 30 35 . . . :

   ```
   overlayTTTTcomment
   ```

   (where T represents a tab character)
6. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines starting with the first character or word that would have gone beyond the truncation column.

Messages

503E	{Truncated|Spilled} [RC=3]
545E	Missing operand(s) [RC=5]
581E	Subcommand is not valid in extended mode [RC=3]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
3	Truncated or spilled, or subcommand is not valid in extended mode
4	No line(s) changed
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

PARSE (Macro)

Use the PARSE macro to help you write new XEDIT macros. The PARSE macro scans a line (of the new macro) that has been transmitted from the console stack to see if its operands match a format specified in the PARSE macro.

Format

PARSE	*startcol*	**Alphaword** **Number** **String** . . . **Dblstring** **Target** **Word** **Line**

Operands

startcol
specifies the starting column in the input line where the parsing is to begin.

The following keywords define the sequence and types of the operands:

Alphaword
specifies that the operand must be an alphabetic character string.

Number
specifies that the operand must be a numeric character string.

String
specifies that the operand must be a delimited string; the delimiter is the first character of the string.

Dblstring
specifies that the operand must be a double string, that is, two adjacent strings separated by a common delimiter. For example, /string1/string2/ is a double string.

Target
specifies that the operand must be an XEDIT target.

Word
specifies that the operand must be a character string, either alphabetic, numeric, or mixed, delimited by blanks.

Line
specifies the unprocessed part of the line being parsed. If the first character is a blank, it is excluded.

Notes for Macro Writers

1. Before issuing the PARSE macro, you must place the line to be parsed in the console stack. For example,

```
push string
```

2. As a result of the PARSE macro, several lines are stacked last-in first-out (LIFO) in the console stack:
   a. The first line contains an integer that indicates the number of stacked lines that follow. One line is stacked for each recognized operand in the parsed line and will be in the order they were specified.
   b. Each of the remaining lines contains the starting column and the length of the operand, except for STRING and DBLSTRING operands.

      A line stacked for a STRING operand contains:

      - the starting column of the delimited string
      - the length of the delimited string (including delimiters)
      - the starting column of the string itself (without the delimiter)
      - the length of the string (without delimiters).

      A line stacked for a DBLSTRING operand contains:

      - the starting column of the delimited strings
      - the length of the delimited strings (including delimiters)
      - the starting column of the first string itself (without the delimiter)
      - the length of the first string (without delimiters)
      - the starting column of the second string (without the delimiter)
      - the length of the second string (without the delimiters).

      Note: For both STRING AND DBLSTRING operands, null strings (explicit or implicit as in // or /) are described as starting in column −1 with a length of 0.

Return Codes

−1	The operands specified in the PARSE macro are incorrect, that is, the first operand is not a number, or one of the subsequent operands is not recognized. Nothing is stacked.
0	Parsing was successful.
1	The scanned line did not match the format specified in the PARSE macro. Parsing was incomplete. Results of the parsing that was completed are available from the console stack.
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

POWERINP

Use the POWERINP (power input) subcommand to enter an input mode in which you can type data as if the screen were one long line. You do not have to be concerned with line length; you can start typing a word on one line of the screen and finish it on the next. When the ENTER key is pressed, the editor divides the data into file lines and puts together any split words.

Format

POWerinp	

Usage Notes

1. When POWERINP is executed, the current file line is displayed in the second line of the screen in protected format, that is, it cannot be modified. The rest of the screen is blank and can be used for input.
2. You can type words continuously and fill up the screen as long as you don't press the ENTER key. When the ENTER key is pressed, the last input line becomes the current line and is displayed at the top of the screen, and you can continue typing data.
3. To exit from power input mode, press the ENTER key without modifying the last displayed screen. The editor divides the data you typed into appropriate file lines. If necessary, lines are cut at word boundaries and, when possible, the data is divided into lines that do not wrap.
4. PF keys cannot be used in power input mode. (Pressing a PF key is equivalent to pressing the ENTER key.) Any prefix subcommands or macros are not executed until you exit from power input mode.
5. If you want to cause a break in the data that you type in power input mode, that is, you want data to start on a new line (for example, a new paragraph or SCRIPT/VS control words, which must start in column one), you can type a line end character (see SET LINEND) before the data that you want to start on a new line. The default line end character is a pound sign (#).

   For example, if the following data is typed in power typing mode:

   ```
   .sp#A pound sign causes the data to start on a new line.#.sp
   ```

   The data will be entered in the file as:

   ```
   ===== .sp
   ===== A pound sign causes the data to start on a new line.
   ===== .sp
   ```

   Any leading blanks after the line end character are eliminated when the data is entered in the file. The first non-blank character after the line end character is placed in column one.
6. A word cannot be longer than the truncation setting.
7. You can use the insert key to insert characters in a line while in power input mode. When characters are inserted, the entire stream of data shifts to the right.

8. Any prefix subcommands or macros are not executed until you exit from power input mode.
9. If the width of your virtual screen or window is less than that of the physical screen, you won't be able to type continuously in power input mode. The keyboard will lock when you come to the end of a line. To unlock the keyboard, press the RESET key and return the cursor to start a new line.

Response

- In power input mode, the screen changes in the following ways:
  - The first line of the screen contains the file ID and the heading,

```
fname ftype fmode * * * P o w e r   T y p i n g * * *  Alt=n
```

  - The second line of the screen contains the current file line in protected format.
  - The rest of the screen is blank.

Messages

117S	Error writing to display terminal [RC=8]
503E	{Truncated|Spilled} [RC=3]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
557S	No more storage to insert lines [RC=4]
581E	Subcommand is not valid in extended mode [RC=3]

Return Codes

0	Normal
3	Truncated or spilled; subcommand valid only for display terminal, or subcommand is not valid in extended mode
4	Insufficient storage available
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	I/O error

PRESERVE

Use the PRESERVE subcommand to save the settings of various XEDIT variables until a subsequent RESTORE subcommand is issued.

Format

PREServe	

Usage Notes

1. The following settings are saved:
   a. Current LEFT or RIGHT
   b. The following SET subcommand options:

```
ARBCHAR     IMPCMSCP    SHADOW
AUTOSAVE    LASTLORC    SPAN
CASE        LINEND      SPILL
CMDLINE     LRECL       STAY
COLOR       MACRO       STREAM
COLPTR      MASK        SYNONYM
CURLINE     MSGMODE     TABLINE
DISPLAY     NULLS       TABS
ESCAPE      NUMBER      TOFEOF
ETARBCH     PACK        TRUNC
FILLER      PREFIX      VARBLANK
FMODE       RECFM       VERIFY
FNAME       SCALE       WRAP
FTYPE       SCOPE       ZONE
HEX         SERIAL      =
IMAGE
```

2. The following values are *not* saved:

   Current line pointer
   Column pointer
   All SET subcommand options not listed above in Usage Note 1b.

Message

520E Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

PURGE

Use the PURGE subcommand to remove the copy of a macro that is in virtual storage.

Format

PURge	*macroname*

Operand

macroname
is the name of a macro, a copy of which is in virtual storage.

Usage Notes

1. When a macro is used for the first time in an editing session, the editor EXECLOADs the macro into virtual storage. The macro remains in storage for the entire session. The macros which the editor has EXECLOADed are EXECDROPed at the end of the session. (When storage becomes unavailable, the copy of the least-recently used macro is erased.) The PURGE subcommand is useful if you modify a macro and want the editor to read the new macro from disk or directory.

   Any time a macro that is being edited is filed or saved, an automatic PURGE is executed for that macro name. Therefore, PURGE is useful only after the CMS commands RENAME or DISK LOAD or with disk operations performed *outside* XEDIT.

2. When a macro invokes the PURGE subcommand and the PURGE is successfully completed, the return code is set to zero. If the macro to be purged is not in storage, the return code is set to 3. If the macro to be purged is in use, the return code is set to 4 and the macro is not purged.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
578W	*macroname* macro is not currently in storage [RC=3]

Return Codes

0	Normal
3	Macro is not currently in storage
4	Macro is in use, do not purge
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

PUT

Use the PUT subcommand to insert one or more lines from the file being edited, starting with the current line, into one of the following: the end of a specified existing file; a new file that you are creating; or a temporary file created by the editor. The original lines remain in the file you are editing.

Format

PUT	[*target* / <u>1</u> [*fn* / = [*ft* / = [*fm* / =]]]]

Operands

target
defines the number of lines to be inserted into another file. Lines are inserted beginning with the current line, up to but not including the target line. If you enter an asterisk (*), the rest of the file is inserted. If you omit target, only the current line is inserted.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

fn
is the file name of the file into which lines are to be inserted. If the file does not exist, the editor creates it and displays the message, Creating new file, in the message area.

ft
is the file type of the file into which lines are to be inserted. If you omit ft, the editor uses the file type of the file you are currently editing.

fm
is the file mode of the file into which lines are to be inserted. If you omit fm, the editor uses the file mode of the file you are currently editing.

Usage Notes

1. The operands fn, ft, or fm may be specified by an equal sign (=), in which case the corresponding value of the file currently being edited is used.
2. If the specified file (fn ft fm) exists, the lines are added to the end of the file. If the file is in an SFS directory, you must have write authority to the file and the directory must be accessed either read/only or read/write.
3. If the specified file (fn ft fm) does not already exist, the editor creates it and inserts the lines. If the file mode indicates an SFS directory, you must have write authority to the directory and it must be accessed either read/only or read/write.
4. If you do not specify a file ID (fn ft fm), the lines specified by target are inserted into a temporary file that the editor creates. These lines can be retrieved each time a subsequent GET subcommand is issued without an operand. This

temporary file is replaced each time a PUT (or PUTD) subcommand is issued. It is erased when XEDIT returns control to the environment which invoked the editor. Thus, issuing a PUT subcommand without specifying a file ID is like storing lines in a temporary holding area. You can retrieve the temporary file without knowing its file ID.

5. If you issue a PUT subcommand without *any* operands, only the current line is placed in a temporary file.
6. When you are editing multiple files, only one temporary file is available. It may be used to insert data from one file into another.

Responses

1. If you are creating a file with the PUT subcommand, the following message is displayed.

```
571I Creating new file:
```

2. The line following the last line PUT (the target line) becomes the new current line.

Messages

037E	Filemode *mode* is accessed as read/only [RC=12]
048E	Invalid filemode *mode* [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
069E	Filemode *mode* not accessed [RC=36]
105S	Error *nn* writing file *fn ft fm* to disk [RC=55, 70, 76, 99, or 100]
229E	Unsupported OS data set, error *nn* [RC=80, 81, 82, 83, or 84]
520E	Invalid operand: *operand* [RC=5]
531E	Disk or file space is full; set new filemode or clear some space [RC=13]
546E	Target not found [RC=2]
554E	Not enough virtual storage available [RC=104]
571I	Creating new file:
579E	Records truncated to *nn* when added to *fn ft fm* [RC=3]
698W	New record length may result in loss of double-byte characters [RC=3]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1141W	User filespace threshold exceeded
1215E	File *fn ft fm* is locked by another user [RC=70]
1258E	Not authorized to write file *fn ft fm* [RC=12]
1262S	Error *nn* opening file *fn ft fm* [RC=55, 70, 76, 99, or 100]
1301S	Rollback error *nn*, file *fn ft fm* left open

Return Codes

0	Normal
1	TOF or EOF reached
2	Target not found
3	Records truncated
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Minidisk is read-only or not authorized
13	Minidisk or file space is full
20	Invalid character in file name or file type
24	Invalid file mode

36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
84	Unsupported OS data set
99	A required system resource is not available
100	Error writing file to disk
104	Insufficient virtual storage

Examples

The following are valid ways to issue the PUT subcommand:

`put`
: (store current line in temporary file, to be inserted by subsequent GET in another file)

`put`*/string/*
: (store from current line up to the line containing string, for use by a subsequent GET in another file)

`put` */string/* `MY NEWFILE`
: (create a new file with lines from current line up to the string)

Refer to the "Examples" section of the GET subcommand for an example of how to use PUT and GET to transfer lines between files while editing multiple files.

PUTD

Use the PUTD subcommand to insert one or more lines from the file being edited, starting with the current line, into one of the following: the end of a specified existing file; a new file that you are creating; or a temporary file created by the editor.

Unlike the PUT subcommand, the PUTD subcommand deletes the original lines from the file you are editing.

Format

PUTD	$\begin{bmatrix} \textit{target} & \begin{bmatrix} \textit{fn} & \begin{bmatrix} \textit{ft} & \begin{bmatrix} \textit{fm} \\ = \end{bmatrix} \\ = & \end{bmatrix} \\ = & \end{bmatrix} \\ \underline{1} & \end{bmatrix}$

Operands

target
defines the number of lines to be inserted into another file. Lines are inserted beginning with the current line, up to but not including the target line. If you enter an asterisk (*), the rest of the file is inserted. If you omit target, only the current line is inserted.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

fn
is the file name of the file into which lines are to be inserted. If the file does not exist, the editor creates it and displays the message, Creating new file, in the message line.

ft
is the file type of the file into which lines are to be inserted. If you omit ft, the editor uses the file type of the file you are currently editing.

fm
is the file mode of the file into which lines are to be inserted. If you omit fm, the editor uses the file mode of the file you are currently editing.

Usage Note

The PUTD subcommand, unlike the PUT subcommand, deletes the original lines from the file.

For additional information, please refer to the "Usage Notes" section of the PUT subcommand.

Responses

If you are creating a file with the PUTD subcommand, the following message is displayed:

```
571I Creating new file:
```

After lines are deleted from the original file, the line immediately following the last line deleted becomes the new current line. If lines are deleted in a backward direction, toward the top of the file, the line preceding the last deleted line becomes the new current line.

Messages

037E	Filemode *mode* is accessed as read/only [RC=12]
048E	Invalid filemode *mode* [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
069E	Filemode *mode* not accessed [RC=36]
105S	Error *nn* writing file *fn ft fm* to disk [RC=55, 70, 76, 99, or 100]
229E	Unsupported OS data set, error *nn* [RC=80, 81, 82, 83, or 84]
520E	Invalid operand: *operand* [RC=5]
531E	Disk or file space is full; set new filemode or clear some space [RC=13]
546E	Target not found [RC=2]
554E	Not enough virtual storage available [RC=104]
559W	Warning: file is empty [RC=1]
571I	Creating new file:
579E	Records truncated to *nn* when added to *fn ft fm* [RC=3]
698W	New record length may result in loss of double-byte characters [RC=3]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1141W	User filespace threshold exceeded
1215E	File *fn ft fm* is locked by another user [RC=70]
1258E	Not authorized to write file *fn ft fm* [RC=12]
1262S	Error *nn* opening file *fn ft fm* [RC=55, 70, 76, 99, or 100]
1301S	Rollback error *nn*, file *fn ft fm* left open

Return Codes

0	Normal
1	TOF or EOF reached
2	Target not found
3	Records truncated
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Minidisk is read-only or not authorized
13	Minidisk or file space is full
20	Invalid character in file name or file type
24	Invalid file mode
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.

81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
84	Unsupported OS data set
99	A required system resource is not available
100	Error writing file to disk
104	Insufficient virtual storage

Examples

See the "Examples" section of the PUT subcommand.

QUERY

Use the QUERY subcommand to display in the message area the current setting of various editing options. Only one option may be specified in each QUERY subcommand.

Format

```
Query    ACTion              NONDisp
         ALT                 NULls
         APL                 NUMber
         ARBchar             PA [n |*]
         AUtosave            PACK
         BASEft              PENDing [BLOCK ][OLDNAME] name |*
         BRKkey              PF [n |*]
         CASE                Point  [*]
         CMDline             PREfix [Synonym *|name]
         COLOR *|field       RANge
         COLPtr              RECFm
         COLumn              REMOte
         CTLchar [char]      RESERved
         CURLine             RING
         CURSor              SCALe
         DISPlay             SCOPE
         EDIRName            SCReen
         EFMode              SELect
         EFName              Seq8
         EFType              SERial
         ENTer               SHADow
         EOF                 SIDcode
         EOL                 SIZe
         ESCape              SPAN
         ETARBCH             SPILL
         ETMODE              STAY
         FILler              STReam
         FMode               SYNonym [*|name]
         FName               TABLine
         FType               TABS
         FULLread            TARGet
         HEX                 TERMinal
         IMage               TEXT
         IMPcmscp            TOF
         LASTLorc            TOFEOF
         LASTmsg             TOL
         LENgth              TRANSLat
         LIBName             TRunc
         LIBType             UNIQueid
         LIne                UNTil
         LINENd              UPDate
         LRecl               VARblank
         LScreen             Verify
         MACRO               VERShift
         MASK                Width
         MEMber              WRap
         MSGLine             Zone
         MSGMode             =
         NBFile
```

Operands

ACTion
displays "ON" or "OFF" to indicate whether any action other than displaying or scrolling has been taken on this file. This includes any file ID change, or file characteristic change (LRECL, RECFM, PACK, SERIAL, SIDCODE, or ALT) as well as any other changes made to the file.

ALT
displays the number of alterations that have been made to the file since the last AUTOSAVE and SAVE, or as specified in the SET ALT subcommand.

APL
displays "ON" or "OFF" as defined by the SET APL subcommand.

ARBchar
displays "ON" or "OFF" and the arbitrary character specified in the SET ARBCHAR subcommand.

AUtosave
displays the current setting defined in the SET AUTOSAVE subcommand: the AUTOSAVE count, file ID, and number of alterations.

BASEft
displays the base file type specified in the LOAD subcommand or the XEDIT command (where the LOAD is implicit).

BRKkey
returns "ON" or "OFF" as defined by the terminal brkkey setting. If ON, then it also returns the PF or PA key currently set to be the BRKKEY.

CASE
displays the case setting "U" or "M" and "R" or "I" as defined in the SET CASE subcommand.

CMDline
displays "ON", "OFF", "TOP", or "BOTTOM" as defined by the SET CMDLINE subcommand.

COLOR *|*field*
displays the current setting defined in the SET COLOR subcommand: field, color, extended highlighting, "HIGH" or "NOHIGH," and programmed symbol set for the field requested or for all fields (* is specified). The fields which may be specified are: Arrow, Cmdline, CUrline, Filearea, Idline, Msgline, Pending, PRefix, Scale, SHadow, STatarea, Tabline, and TOfeof.

COLPtr
displays "ON" or "OFF" as defined by the SET COLPTR subcommand.

COLumn
displays the column number of the column pointer.

CTLchar *[char]*
If no character is specified (Q CTLCHAR), displays the escape character and all control characters defined by the SET CTLCHAR subcommand, in the form "CTLCHAR ON ESCAPE char CTL c1 c2 c3 c4...." If no control characters have been defined, displays "CTLCHAR OFF."

If a character is specified (Q CTLCHAR char), the attributes of that character are displayed, in the form "CTLCHAR char attribute1 (PROTECT|NOPROTECT), attribute2 (color), attribute3 (extended highlighting), attribute4 (HIGH|NOHIGH|INVISIBLE), attribute5 (PSs)." If no attributes were defined for the character, displays "CTLCHAR char."

CURLine
displays the line number of the current line as specified by the SET CURLINE subcommand.

CURSor
displays the current position of the cursor on the screen (line number and column number), and the position of the cursor in the file (line number and column number). If the cursor is not in the file area, two negative numbers (−1) are displayed for the position of the cursor in the file. The top and bottom of the range are considered to be in the file.

The current position of the cursor is the location where the cursor would be placed if the screen were displayed at this time. The current position reflects relative changes due to additions/deletions of lines resulting from prefix execution. It does not necessarily reflect where the cursor will eventually be located when the screen is actually displayed. This is because cursor priority cannot be resolved until the screen is displayed.

DISPlay
displays the range of selection levels which are included in the display, as specified by the SET DISPLAY subcommand.

EDIRName
displays the name of the SFS directory containing the file at the time the file was first loaded.

EFMode
displays the two-character file mode of the file at the time the file was first loaded.

EFName
displays the eight-character file name of the file at the time the file was first loaded.

EFType
displays the eight-character file type of the file at the time the file was first loaded.

ENTer
displays "BEFORE", "AFTER", "ONLY", or "IGNORE" and the ENTER key definition as set by the SET ENTER subcommand.

EOF
displays "ON" or "OFF" as determined by the editor. EOF is "ON" when the line pointer reaches end of file (or end of range).

EOL
displays "ON" or "OFF" as determined by the editor. EOL is "ON" when the column pointer reaches zone2 + 1.

ESCape
displays "ON" or "OFF" and the escape character defined by the SET ESCAPE subcommand.

ETARBCH
displays "ON" or "OFF" and the extended arbitrary character defined by the SET ETARBCH subcommand.

ETMODE
displays "ON" or "OFF" as defined by the SET ETMODE subcommand.

FILler
displays the filler character defined by the SET FILLER subcommand.

FMode
displays the two-character file mode.

FName
displays the eight-character file name.

FType
displays the eight-character file type.

FULLread
displays "ON" or "OFF" as defined by the SET FULLREAD subcommand.

HEX
displays "ON" or "OFF" as specified in the SET HEX subcommand.

IMage
displays "ON," "OFF," or "CANON" as specified in the SET IMAGE subcommand.

IMPcmscp
displays "ON" or "OFF" as specified in the SET IMPCMSCP subcommand.

LASTLorc
displays the current contents of the last locate or change buffer, as specified either by SET LASTLORC or by the editor.

LASTmsg
displays the last message issued by the editor. This message may or may not have been displayed, depending on the SET MSGMODE subcommand operands.

LENgth
displays the length of the current line from column one through the truncation column (excluding trailing blanks). The length is zero for Top of File and End of File lines.

LIBName
displays "LIBNAME filename" when MEMBER is ON. The file name is the library file name specified on the XEDIT or LOAD command. When MEMBER is OFF, the library file name is blank.

LIBType
displays "LIBTYPE filetype" when MEMBER is ON. The file type is the library file type specified on the XEDIT or LOAD command. When MEMBER is OFF, the library file type is blank.

LIne
displays the current line number, relative to the beginning of the file.

LINENd
displays "ON" or "OFF" and the line end character as defined by the SET LINEND subcommand.

LRecl
displays the logical record length of the file.

LScreen
displays six integers:

- The number of lines and the number of columns in the logical screen
- The line number and the column number defining the top left corner of the logical screen on the virtual screen
- The number of lines and number of columns in the virtual screen.

MACRO
displays "ON" or "OFF" as specified by the SET MACRO subcommand.

MASK
displays the current mask line as defined by the SET MASK subcommand.

MEMber
displays "MEMBER ON" when editing a member of a CMS library or "MEMBER OFF" when not editing a library member.

MSGLine
displays "ON" or "OFF," the location of the message line, the number of lines the message line can expand to, and OVERLAY, if that has been specified, as defined by the SET MSGLINE subcommand.

MSGMode
displays "ON" and "LONG" or "SHORT" as defined by the SET MSGMODE subcommand, or, if SET MSGMODE OFF has been issued, nothing is displayed.

NBFile
displays the number of files you are currently editing.

NONDisp
displays the character defined by the SET NONDISP subcommand.

NULls
displays "ON" or "OFF" as specified by the SET NULLS subcommand.

NUMber
displays "ON" or "OFF" as specified by the SET NUMBER subcommand.

PA [*n*|<u>*</u>]
displays "BEFORE," "AFTER," "ONLY," or "IGNORE" and:

1. If you specify an asterisk or no argument at all, QUERY PA displays all PA key definitions.
2. If you specify a particular key (PA*n*), QUERY PA displays that key's setting.

PACK
displays "ON" or "OFF" as specified by the SET PACK subcommand.

PENDing [BLOCK] [OLDNAME] ***name*|***
displays the first entry in the pending list with "name" name or all the entries in the pending list (if * specified).

BLOCK

indicates that the pending list is to be checked for BLOCK entries only.

OLDNAME

indicates that the name specified is the original name of the prefix subcommand or macro.

name

indicates the name of the prefix subcommand or macro for which you are searching. If OLDNAME is also specified, name must be the original name of the prefix subcommand or macro, regardless of whether or not a synonym has been assigned to that name. Otherwise it is assumed to be a synonym (that is, a new name) or a name without a synonym.

*

indicates to display all of the entries in the pending list. If BLOCK is also specified, * indicates to display only the block entries.

The information is returned in the following form:

```
Line n : 'name', Oldname='name', OP1='x', OP2='y', OP3='z'
```

PF [*n*|*]

displays "BEFORE", "AFTER", "ONLY", or "IGNORE" and:

1. If you specify an asterisk or no argument at all, QUERY PF displays all PF key definitions.
2. If you specify a particular key (PF*n*), QUERY PF displays that key's setting.

Note: If you are editing a file in line mode (SET TERMINAL TYPEWRITER), XEDIT does not recognize which key caused an attention interrupt. The definition which will be executed will be the CP definition of that key. To QUERY keys in line mode, use the CP QUERY PF*nn* command.

Point [*]

QUERY POINT displays the symbolic name(s) associated with the current line, or displays a blank string if there are no names specified for this line. QUERY POINT * displays all symbolic names that have been defined, starting at the top of the file including the line number to which each name applies.

PREfix [Synonym *|*name*]

QUERY PREFIX displays "ON", "OFF", or "NULLS" and "RIGHT" or "LEFT" as specified in the SET PREFIX subcommand.

QUERY PREFIX SYNONYM * displays both the old and the new names of the synonyms defined for the prefix subcommand(s) or macro(s).

QUERY PREFIX SYNONYM *name* displays the synonym for the specified prefix subcommand or macro and its associated old name.

RANge

displays the line numbers of the top and bottom of the range defined by the SET RANGE subcommand.

RECFm

displays the record format, "F", "V", "FP", or "VP", defined by the SET RECFM subcommand.

REMOte

displays "ON" or "OFF" depending upon whether or not a remote terminal is being used or upon the setting specified by the SET REMOTE subcommand.

RESERved

displays (on one line) the line numbers of the screen lines currently reserved. The line numbers are displayed in the order that they were reserved.

RING

displays the number of files you are editing and the file identification line for each file.

The editor displays the following message:

```
530I nn file(s) in storage
```

The following information is displayed for each file:

```
fn ft fm recfm lrecl Trunc=truncno Size=sizeno
Line=lineno Col=colno Alt=altcount
```

where:

`fn`
: is the file name of the file.

`ft`
: is the file type of the file.

`fm`
: is the file mode of the file.

`recfm`
: is the record format. F is fixed-length records. V is variable-length records. FP is fixed-length packed. VP is variable length packed.

`lrecl`
: the logical record length of the largest record permitted in the file.

`truncno`
: is the truncation column.

`sizeno`
: is the number of records currently in the file.

`lineno`
: is the position in the file of the current line.

`colno`
: is the column number in which the column pointer is located.

`altcount`
: is the number of alterations since the last autosave or the number set by SET ALT.

SCALe

displays "ON" or "OFF," and the position of the SCALE as specified in the SET SCALE subcommand (or SCALE prefix subcommand).

SCOPE

displays "DISPLAY" or "ALL" as specified by the SET SCOPE subcommand.

SCReen
displays the attributes of the screens that have been defined by the SET SCREEN subcommand preceded by SIZE, WIDTH, or DEFINE.

SELect
displays the selection level of the current line and the maximum selection level for the file as specified by the SET SELECT subcommand.

Seq8
displays "OFF" if the XEDIT command or LOAD subcommand was issued with the NOSEQ8 operand; if not, displays "ON."

SERial
displays the serial identification, the increment value, and the serial number starting value as defined by the SET SERIAL subcommand.

SHADow
displays "ON" or "OFF" as specified by the SET SHADOW subcommand.

SIDcode
displays the eight-character string specified in the SIDCODE option of the XEDIT command, the LOAD subcommand, or the SET SIDCODE subcommand.

SIZe
displays the number of records in the file being edited.

SPAN
displays "ON" or "OFF," "B" or "N," and n as defined in the SET SPAN subcommand.

SPILL
displays "ON," "OFF," or "WORD" as defined by the SET SPILL subcommand.

STAY
displays "ON" or "OFF" as specified in the SET STAY subcommand.

STReam
displays "ON" or "OFF" as specified in the SET STREAM subcommand.

SYNonym [*|*name*]
QUERY SYNONYM displays "ON" or "OFF" as specified in the SET SYNONYM subcommand.

QUERY SYNONYM * displays for each defined synonym its name, its minimum abbreviation, and the associated synonym definition which includes the LINEND char, if specified, and everything that was specified in the SET SYNONYM subcommand after the size of the minimum abbreviation.

QUERY SYNONYM *name* displays the synonym, its minimum abbreviation, and the associated synonym definition which includes the LINEND char, if specified, and everything that was specified in the SET SYNONYM subcommand after the size of the minimum abbreviation.

(If no synonym was defined, only the name is displayed.) For example:

```
set synonym up 1 down
query syn u
```

displays the following: `SYNONYM UP U DOWN`

TABLine
displays "ON" or "OFF" and the position of the TABLINE as defined in the SET TABLINE subcommand (or TABL prefix subcommand).

TABS
displays the tab column numbers defined by the SET TABS subcommand.

TARGet
displays the following information about the character string that matches the last target located via a LOCATE or CLOCATE subcommand: line and column number of the first character in the string; line and column number of the last character in the string. If the last target located was specified with "&," then only information about the last string found is displayed.

If the target of the LOCATE or CLOCATE has been specified as an absolute line number, a relative displacement from the current line, or a line name, then QUERY TARGET displays the line number and current column position (twice).

Information returned by QUERY TARGET may be incorrect unless the QUERY is done immediately following the LOCATE or CLOCATE of the target. Any XEDIT subcommand issued between the LOCATE or CLOCATE of the target and the QUERY has the potential to invalidate the TARGET information.

TERMinal
displays "DISPLAY" or "TYPEWRITER" as defined in the SET TERMINAL subcommand.

TEXT
displays "ON" or "OFF" as specified in the SET TEXT subcommand.

TOF
displays "ON" or "OFF" as determined by the editor. TOF is "ON" when the line pointer reaches top of file (or top of range).

TOFEOF
displays "ON" or "OFF" as specified in the SET TOFEOF subcommand.

TOL
displays "ON" or "OFF" as determined by the editor. TOL is "ON" when the column pointer reaches zone1 − 1.

TRANSLat
displays "ON" or "OFF," depending on whether or not the user has defined pairs of uppercase translate characters using the SET TRANSLAT subcommand.

TRunc
displays the truncation column number as defined by the SET TRUNC subcommand.

UNIQueid
displays a unique identifier associated with the file. The identifier has the form *rrrnnnnn* where *rrr* is the number of times XEDIT was called recursively and *nnnnn* is the current autosave number. Note that when the recursion level, *rrr*, is less than 100, the leading zero(es) will be dropped. The uniqueid is also used as the file name for the AUTOSAVE file.

UNTil
displays the file type up through which updates were applied as specified on the XEDIT command or LOAD subcommand.

UPDate
displays "ON" or "OFF" as determined by the editor. Update is "ON" when the XEDIT command or LOAD subcommand has been issued and the UPDATE option was specified or implied.

VARblank
displays "ON" or "OFF" as specified in the SET VARBLANK subcommand.

Verify
displays the verification columns and "ON" or "OFF" as specified in the SET VERIFY subcommand.

VERShift
displays n or −n, which is the relative position of the screen over the file, as a result of any LEFT or RIGHT subcommands.

Width
displays the WIDTH value specified in the XEDIT command or LOAD subcommand.

WRap
displays "ON" or "OFF" as specified in the SET WRAP subcommand.

Zone
displays the left and right zone column numbers specified in the SET ZONE subcommand.

=

displays the string in the equal (=) buffer. The = buffer contains the last executed subcommand or macro or CP/CMS command, or whatever has been specified in the SET = subcommand.

Messages

520E	Invalid operand: *operand* [RC=5]
525E	Invalid {PFkey|PFkey/PAkey} number [RC=5]
538E	No name defined [RC=3]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
3	'QUERY POINT *' was issued, but no symbolic names are defined.
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

QUIT

Use the QUIT subcommand to terminate the current editing session and leave the previous copy of the file, if any, intact on the disk or directory. If the file has been changed during the editing session, the editor displays a warning message asking you to confirm that you want to issue a QUIT (instead of a FILE) subcommand. When issued from a macro, the QUIT subcommand can be used to specify a return code.

Format

QUIT	[*n*]

Operand

n
is a return code that may be specified when QUIT is issued from a macro.

Usage Notes

1. You can use the QUIT subcommand when you edit a file merely to examine, but not to change, its contents, or whenever you discover you have made errors in editing a file and want to cancel your editing session.
2. If only one file was edited, control is returned to the environment which invoked the editor.
3. If more than one file was being edited, only the current file is terminated. Control remains in the edit environment. You can use the CANCEL macro to "quit" multiple files.
4. The QUIT subcommand is initially assigned to the PF3 key.
5. When editing two or more members of a MACLIB in member mode, the lock on the library is not deleted until the last member that had the lock option in effect exits the ring.

Notes for Macro Writers

1. QUIT is defined as a synonym to PQUIT. The PQUIT (protected quit) subcommand causes a warning message to be displayed (see responses) if the file was changed during editing. To bypass the message and have the QUIT subcommand executed directly, you can define QUIT as a synonym to COMMAND QUIT, or you can issue QQUIT, which is an unprotected "quick quit."

   The PQUIT subcommand clears the console stack. If you do not want the console stack to be cleared, use COMMAND QUIT.

2. If QUIT is issued from a macro, control remains in the macro until the macro finishes executing. Then, control is returned to the editor.

   If multiple files were being edited, only the current file is terminated.

   If QUIT executes successfully from a macro, the return code is set to *n* if specified. If *n* was not specified and only one file was being edited, a QUIT issued from a macro sets the return code to 1 and executes the QUIT when the macro completes execution. After issuing the QUIT, any subcommands issued in the macro will result in a return code of 6.

3. QUIT or PQUIT may not be issued from a prefix macro.
4. Negative numbers may not be specified for *n*.

Responses

If only *one* file was edited, the CMS ready message indicates that control has been returned to CMS. If more than one file was being edited, the file that was being edited before the terminated file appears on the screen.

If the file was changed during the editing session, the following message is displayed:

```
577E File has been changed; use QQUIT to quit anyway
```

If you wish to save the changes, issue a FILE subcommand.

On a typewriter terminal, the following message is displayed:

```
553I Editing file: fn ft fm
```

Messages

509E	*subcommand* subcommand not valid from a prefix macro [RC=4]
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
553I	Editing file: *fn ft fm*
554E	Not enough virtual storage available
577E	File has been changed; type QQUIT to quit anyway [RC=12]
622E	Insufficient free storage for MSGLINE
1300E	Error *nn* unlocking file *fn ft dirname*

Return Codes

0	Normal
1	Only one file was edited
4	Invalid when issued from a prefix macro
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	File has been changed. Use QQUIT to QUIT anyway
n	The number specified as an operand to the subcommand

READ

Use the READ subcommand to place information from the terminal in the console stack (LIFO). The READ subcommand is designed to be issued from a macro. After a READ subcommand is executed, a System Product Interpreter macro has access to the stacked information via the REXX PULL instruction. (In an EXEC 2 macro, access can be gained via the &READ control statement).

Format

```
READ     ┌ Cmdline               ┐  ┌ Tag   ┐
         │ All  [Number]         │  │       │
         └ Nochange  [Number]    ┘  └ Notag ┘
```

Operands

<u>**Cmdline**</u>
: indicates that only the command input area is to be stacked in the console stack. This is the default.

All
: indicates that all lines changed on the screen will be stacked in the console stack, and the command input area will be stacked as the last line. The corresponding changes will also be made to the copy of the file that is in virtual storage.

Nochange
: is similar to the ALL option (above) except that the changes made on the screen are not made to the file being edited but are available from the console stack.

Number
: indicates that the lines changed on the screen are to be prefixed by their file line numbers.

Tag
: specifies that a tag identifying the origin of the line is inserted at the beginning of each line stacked. (The tags are described in note 5, below.)

<u>**Notag**</u>
: specifies that no tags are inserted in the stacked lines.

Notes for Macro Writers

1. If the console stack already contains a line when a READ subcommand is issued, the READ is a no-op (no operation takes place).
2. If a READ is issued from a typewriter terminal, only the command line is placed in the stack.
3. A READ subcommand is terminated by pressing either the ENTER key, a PF key, or a PA key; however, a key assigned to COPYKEY, NULLKEY, TABKEY, or the CP BRKKEY does not terminate a READ. Pressing the CLEAR key clears any data that was typed since the last time the ENTER key, a PF key, or a PA key was pressed, but it also does not terminate a READ. The operand specified for the READ subcommand (Cmdline, All, or Nochange) determines which lines are placed in the console stack. The same number of

lines are stacked whether TAG or NOTAG is specified. The TAG operand causes each line in the stack to be preceded by a tag. The various tags are described in note 4, below.

Using READ CMDLINE

The key pressed to terminate the READ command (a PF key, a PA key, or the ENTER key) and how that key is defined (BEFORE, AFTER, ONLY, or IGNORE), determine which lines are placed in the console stack.

If something was entered on the command line, the console stack will contain the following line(s), as determined by the definition of the key that was pressed:

If the key was defined as 'BEFORE', the console stack will contain:

a. ENTER, PF, or PA key value (depending on which key was pressed)
b. Command line.

If the key was defined as 'AFTER', the console stack will contain:

a. Command line
b. ENTER, PF, or PA key value (depending on which key was pressed).

If the key was defined as 'IGNORE', the console stack will contain:

a. Command line.

If the key was defined as 'ONLY', the console stack will contain:

a. ENTER, PF, or PA key value (depending on which key was pressed).

If nothing was entered on the command line (or ERASE EOF was used to clear the command line), the console stack will contain the definition of the key (ENTER, PF, or PA key) that was pressed.

Using READ ALL or READ NOCHANGE

The editor scans the screen (from top to bottom, left to right) and stacks all modified fields. Each modified field is stacked as a separate line. If either the ALL or NOCHANGE operand is used, the console stack contains:

a. ENTER, PF, or PA key value (depending on which key was pressed)
b. Lines and prefix areas changed on the screen (if any)
c. Command line (if something was entered on the command line).

If nothing was changed on the screen, the console stack contains the ENTER, PF, or PA key value (depending on which key was pressed).

4. With the TAG operand specified, each line in the stack is preceded by a tag, which identifies the field. The tags and their meanings are as follows:

CMD	– command line
ETK	– ENTER key
FIL	– file line
PAK	– PA key
PFK	– PF key
PRF	– prefix area
RES	– reserved line

The tag is followed by additional information and the modified field itself:

- CMD string

  where string is whatever was typed in the command line.

The string can be empty if a key which is not defined is pressed. In this case, only the tag (CMD) is stacked (or a null line if READ NOTAG is specified).

- ETK string

  where string is the ENTER key definition.

- FIL n1 n2 [n3] string

  where:

  `n1 n2`
  : are the line number and column number of the beginning of the line on the screen.

  `n3`
  : is the corresponding file line number. The file line number is returned only if the READ subcommand was issued with the NUMBER option.

  `string`
  : is the changed file line. (String can be empty if the ERASE EOF key was pressed.)

- `PAK n string`

  where:

  n
  : is the number of the PA key that was pressed to terminate the READ.

  `string`
  : is the PA key definition.

  PAK lines are stacked LIFO.

- `PFK n string`

  where:

  `n`
  : is the number of the PF key that was pressed to terminate the READ.

  `string`
  : is the PF key definition.

  PFK lines are stacked LIFO.

- PRF n1 n2 [n3] string

  where:

  `n1 n2`
  : are the line number and column number of the prefix area on the screen.

  `n3`
  : is the corresponding file line number associated with the prefix area. The file line number is returned only if the READ subcommand was issued with the NUMBER option.

`string`
: is whatever was typed in the prefix area. (String can be empty if the ERASE EOF key was pressed). The string is whatever the user typed in the prefix area, as determined by XEDIT. For example, with SET NUMBER OFF, typing "a" in the prefix area returns "a" and not "a====".

- `RES n1 n2 string`

  where:

  `n1 n2`
  : are the line number and column number of the reserved field on the screen.

  `string`
  : is the reserved field that was changed. (String can be empty if the ERASE EOF key was pressed.)

5. If a command is entered on a screen that is in "MORE" status, the command is placed in the CMS input queue.

Responses

The message, "Macro-read *n* File(s)", is displayed in the status area.

Messages

509E	*subcommand* subcommand not valid from a prefix macro [RC=4]
520E	Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
1	TOF or EOF reached
4	Invalid when issued from a prefix macro
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

RECOVER

Use the RECOVER subcommand to replace a specified number of lines that were removed by a DELETE, a MERGE, or a PUTD subcommand, or a D (delete) prefix subcommand.

Format

RECover	[*n* | * | <u>1</u>]

Operand

n

is the number of lines removed by a DELETE, a MERGE, or a PUTD subcommand or a D prefix subcommand that you wish to replace in the file. If you specify an asterisk (*), all deleted lines are replaced. If you omit n, only the last line that was removed is reinserted in the file.

Usage Notes

1. The last lines that were deleted are the first to be recovered. For instance, a DELETE 10 followed by a RECOVER 5 would recover the *last* five lines of the 10 deleted lines.
2. Once a deleted line is recovered, it is removed from the buffer that contains recoverable lines. Therefore, DELETE followed by multiple RECOVER subcommands cannot be used to duplicate a line at various locations in a file.
3. When multiple files are being edited, there is only *one* buffer for all removed lines. Thus, a line could be deleted in one file and recovered in another file if the width of the two files is the same.
4. RECOVER subcommands do *not* have to match DELETE subcommands. The following is a valid sequence:

```
delete 2
  .
  .
  .
delete 3
  .
  .
  .
recover 1
  .
  .
  .
recover 4
```

5. The size of the recoverable lines buffer depends on the amount of virtual storage.
6. Macros, such as JOIN, that use the DELETE, MERGE or PUTD subcommands will put lines in the buffer that contains recoverable lines.
7. If you are editing a file in update mode (where the status area indicates Update-mode n File(s)), deleted lines that are reflected in an update file cannot

be recovered. Please refer to the XEDIT command in this publication for more information on update mode.

Response

The recovered line(s) is inserted at the current line. The following message is displayed:

```
502I nn line(s) recovered
```

The recovered line becomes the new current line.

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
502I	{No|*nn*} line(s) recovered [RC=0 or 3]

Return Codes

0	'n' lines have been inserted
3	Pool of deleted lines is empty
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

REFRESH

Use the REFRESH subcommand to display the screen. Issued from a macro, it presents the screen as of that moment in processing, without waiting for input.

Format

REFRESH	

Note to Macro Writers

REFRESH can be used to update the display on the screen without exiting the macro.

Messages

117S	Error writing to display terminal [RC=8]
520E	Invalid operand: *operand* [RC=5]
529E	Subcommand is only valid in {display|editing} mode [RC=3]

Return Codes

0	Normal
3	Subcommand valid only for display terminal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	I/O error writing to screen

RENUM

Use the RENUM subcommand to renumber the line numbers of VSBASIC or FREEFORT files.

Format

RENum	$\left[\begin{matrix} \textit{startno} \\ \underline{\mathbf{10}} \end{matrix} \left[\textit{incr}\right]\right]$

Operands

startno
: specifies the first line number of the file. If you omit startno, the file starts with line number 10.

incr
: specifies the value by which each line number is incremented. If you omit incr, the value specified as startno is assumed.

Usage Notes

1. The startno and incr values cannot exceed 99999 for VSBASIC files or 99999999 for FREEFORT files.
2. This subcommand is intended to be used in EDIT migration mode; XEDIT does not perform line number editing.
3. This subcommand should not be used for a VSBASIC file that has a logical record length greater than 256.

Responses

If you are not in a VSBASIC or FREEPORT file and you enter the RENUM command, the following message is displayed:

```
Wrong file format for RENUM
```

When RENUM finds an invalid VSBASIC or FREEFORT statement in your file, one of the following messages is displayed:

```
Maximum line number exceeded
Overflow at statement number
Invalid syntax in statement number
Invalid line number reference in statement number
Line number referenced in statement number not found
```

Messages

002E	File *fn ft fm* not found [RC=28]
037E	Filemode *mode* is accessed as read/only [RC=12]
069E	Filemode *mode* not accessed [RC=36]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, 70, 99, or 100]
105S	Error *nn* writing file *fn ft fm* on disk [RC=31, 55, 70, 99, or 100]
520E	Invalid operand: *operand* [RC=5]
531E	Disk is full; set new filemode or clear some disk space [RC=13]
535E	Invalid parameters for RENUM [RC=5]
543E	Invalid number: *number* [RC=5]
554E	Not enough virtual storage available [RC=104]

Return Codes

0	Normal
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	File mode is accessed as read/only
13	Minidisk or file space is full
28	File not found
31	A rollback occurred
32	Format or content of input file invalid
36	Mode not accessed
55	APPC/VM communications error
70	File sharing conflict
99	A required system resource is not available
100	Error reading/writing file to disk
104	Out of virtual storage

REPEAT

Use the REPEAT subcommand to advance the line pointer and to execute the last subcommand that was entered. The REPEAT subcommand is equivalent to executing the two subcommands, NEXT 1 (or UP 1) and =, for a specified number of times.

Format

REPEat	[*target* <u>1</u>]

Operand

target
specifies the number of times that the REPEAT subcommand executes, starting on the line following the current line. REPEAT computes the number of lines between the current line and the line preceding the target line. The last subcommand entered is then REPEATed for this number of times. If you specify an asterisk (*), the previous subcommand is repeated up to the end of file. If you omit target, the previous subcommand is executed on the line following the current one.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Notes

1. If the target is in a forward direction, REPEAT is equivalent to:

   ```
   next 1
   =
   ```

2. If the target is in a backward direction, that is, specified with a leading minus (−) sign, REPEAT is equivalent to:

   ```
   up 1
   =
   ```

3. If the previous subcommand contains a target or is a subcommand whose purpose is to move the line pointer (such as NEXT), the execution of the REPEAT subcommand may occur beyond the line specified by the target of the REPEAT.

4. If the previous subcommand results in a non-zero return code, the execution of the REPEAT subcommand will terminate on that line.

Response

The response, if any, from the repeated subcommand is displayed.

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]

Return Codes

1	TOF or EOF reached
2	Target not found
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
nn	Same as the repeated subcommand's return codes

REPLACE

Use the REPLACE subcommand to replace the current line with a specified line or to delete the current line and enter input mode.

Format

Replace	[*text*]

Operand

text
specifies an input line that is to replace the current line. If a line is specified, the editor puts it into the file in place of the current line. If no line is specified, the editor deletes the current line and enters input mode.

Usage Notes

1. If a REPLACE subcommand is issued when the current line is the Top of File line, the text is inserted after the Top of File line. If it is issued when the current line is the End of File line, the text is inserted before the End of File line.
2. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines starting with the first character or word that would have gone beyond the truncation column.
3. If REPLACE is issued without operands, the current line is deleted and input mode is entered. If you do not enter any data, the deleted line is recovered. However, if the editor is in update mode the deleted line is not recovered.
4. If SET IMAGE ON is in effect, tabs are converted to blanks before a line is inserted into the file.

Responses

When you issue a REPLACE subcommand with no operand, the following message is displayed:

```
573I Input mode:
```

Messages

503E	{Truncated|Spilled} [RC=3]
557S	No more storage to insert lines [RC=4]

Return Codes

0	Normal
3	Truncated or spilled
4	Not enough storage to add lines
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line

```
===== This is the current line.
```

```
replace Now this is the new current line.
```

```
===== Now this is the new current line.
```

RESET

Use the RESET subcommand to remove all prefix subcommands or macros when the screen is in a "pending" status.

Format

RESet	

Usage Notes

1. The RESET subcommand removes all prefix subcommands and/or macros when the screen displays a pending status.
2. To remove prefix subcommands or macros when the screen is not in a pending status, press the CLEAR key.
3. RESET does not reset the prefix area of shadow lines or any lines outside the defined scope. See SET SCOPE.

Note for Prefix Macro Writers

If a user enters RESET when a prefix macro is pending, that macro is invoked with the argument string "PREFIX CLEAR pline" where "pline" is the line number for the line on which the prefix macro was pending. When the macro examines the argument string and finds it was invoked with the CLEAR argument, it would (usually) stop processing. (For a complete description of the argument string that is automatically passed to a prefix macro, see the *VM/SP System Product Editor User's Guide*.)

Response

The original prefix area (equal signs or line numbers) replaces any prefix subcommands or macros that were typed in.

Message

520E Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

If a block of lines to be moved or copied is initiated by typing MM or CC in the first and last lines of the block, but the destination line has not yet been entered (via P or F), the operation can be canceled by RESET.

RESTORE

Use the RESTORE subcommand to restore the settings of the XEDIT variables to the values they had when the PRESERVE subcommand was last issued.

Format

RESTore	

Usage Notes

1. Refer to the PRESERVE subcommand for a list of the variables affected by the RESTORE subcommand.
2. If the column pointer was located outside the restored zone settings, it is repositioned to the left or right zone. If the column pointer was positioned to the left of the new zone, it moves to Top of Line (TOL). If it was positioned beyond the new right zone, it moves to End of Line (EOL).

Messages

507E	No preserved data to restore [RC=3]
520E	Invalid operand: *operand* [RC=5]

Return Codes

0	Normal
1	The column pointer was located outside the restored zone settings
3	No PRESERVE has been issued
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

RGTLEFT (Macro)

Use the RGTLEFT macro to view columns of data that are not currently visible on the screen.

Format

RGTLEFT	[*n*]

Operand

n

is the number of columns that you want the screen to be moved. Whether the screen moves to the right or the left depends on the current VERSHIFT value (see usage note 3). If n is not specified, the screen is moved up to a maximum of three-fourths of the logical screen width (for example, 60 characters on an 80-character screen). If the logical record length is less than this value (screen width plus 3/4 screen width), the screen moves only enough to display the rest of the logical record, up to a maximum of three-fourths of the first verify width (the number of columns in the first verify pair). For example, if VERIFY is set at 1 40 on a file with a record length of 80, the first RGTLEFT command moves the screen right 30 columns (3/4 of 40) and columns 31 through 70 are displayed.

Usage Notes

1. The RGTLEFT macro does not cause data to be lost, nor does it move the line or column pointer.
2. The RGTLEFT macro is most useful when assigned to a PF key. The PF10 key is initially set to RGTLEFT. Pressing the PF key enables you to see data to the right of the screen; pressing the PF key again returns the screen to the original view of the file.
3. The current VERSHIFT setting (see QUERY VERSHIFT) determines whether the view is moved right or left when RGTLEFT is issued. If VERSHIFT is less than or equal to zero, the view is moved to the right, making the VERSHIFT value greater than zero. When VERSHIFT is greater than zero, the view is moved back to the left, returning to the original display of the file.

Messages

520E	Invalid operand: *operand* [RC = 5]
529E	RGTLEFT is only valid in {display|editing} mode [RC = 3]
543E	Invalid number: *number* [RC = 5]
576E	{Total verify width exceeds screen size (*nn*)|Total offset exceeds LRECL (*nn*)} [RC = 5]

Return Codes

0	Normal
3	RGTLEFT valid in display mode only
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

RIGHT

Use the RIGHT subcommand to view columns of data that are not currently visible on the screen. The RIGHT subcommand allows you to see data that is *to the right of the last* (right-most) *column* on the screen. The data moves to the left, thus allowing you to see a specified number of positions to the right of the last column.

Format

RIght	[*n* |1]

Operand

n

specifies the number of positions to the right of the last column on the screen that you want to see. If you omit n, one position to the right of the last column becomes visible.

Usage Notes

1. The RIGHT subcommand does not cause data to be lost, nor does it move the line or column pointer.
2. To get the data back to its original position, use the LEFT n subcommand. See also the RGTLEFT macro.
3. RIGHT subcommands are cumulative. For example:
   ```
   ri 10
   ri 10
   ```
   is equivalent to
   ```
   ri 20
   ```
4. If you have issued several RIGHT and/or LEFT subcommands and have forgotten the total value of n:
   a. RIGHT 0 or LEFT 0 restores the screen to its original display.
   b. SET VERIFY resets RIGHT (or LEFT) to zero.
   c. QUERY VERSHIFT displays n (the result of a RIGHT) or −n (the result of a LEFT).
5. The total number of columns moved cannot exceed the logical record length.

Notes for Macro Writers

EXTRACT/VERSHIFT/ returns the value of n or −n.

Response

The screen moves to the right relative to the file data.

Messages

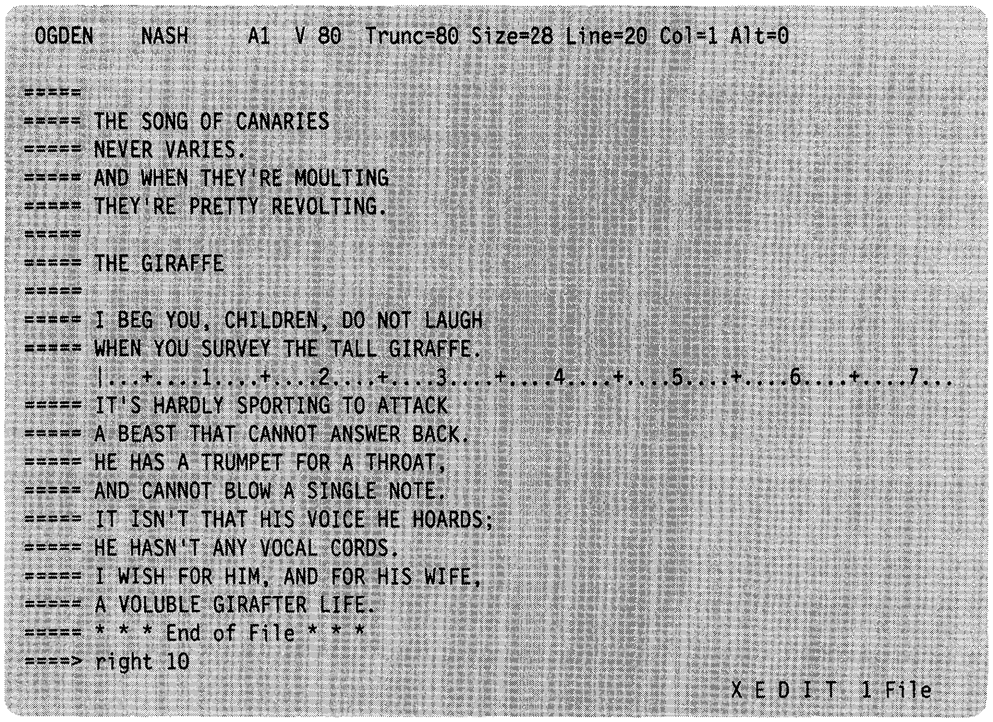
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
576E	{Total verify width exceeds screen size (*nn*)|Total offset exceeds LRECL (*nn*)} [RC=5]

Return Codes

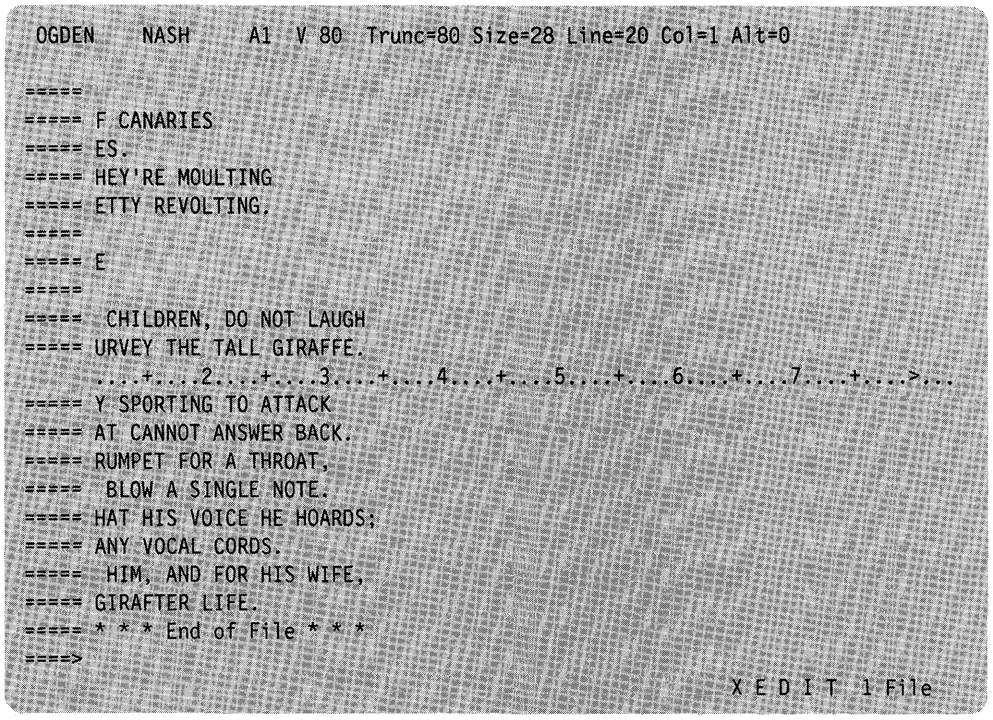
0	Normal
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Figure 12 is a before-and-after example of the RIGHT subcommand.



```
 OGDEN    NASH     A1  V 80  Trunc=80 Size=28 Line=20 Col=1 Alt=0

=====
===== THE SONG OF CANARIES
===== NEVER VARIES.
===== AND WHEN THEY'RE MOULTING
===== THEY'RE PRETTY REVOLTING.
=====
===== THE GIRAFFE
=====
===== I BEG YOU, CHILDREN, DO NOT LAUGH
===== WHEN YOU SURVEY THE TALL GIRAFFE.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== IT'S HARDLY SPORTING TO ATTACK
===== A BEAST THAT CANNOT ANSWER BACK.
===== HE HAS A TRUMPET FOR A THROAT,
===== AND CANNOT BLOW A SINGLE NOTE.
===== IT ISN'T THAT HIS VOICE HE HOARDS;
===== HE HASN'T ANY VOCAL CORDS.
===== I WISH FOR HIM, AND FOR HIS WIFE,
===== A VOLUBLE GIRAFTER LIFE.
===== * * * End of File * * *
====> right 10
                                                      X E D I T  1 File
```



```
 OGDEN    NASH     A1  V 80  Trunc=80 Size=28 Line=20 Col=1 Alt=0

=====
===== F CANARIES
===== ES.
===== HEY'RE MOULTING
===== ETTY REVOLTING.
=====
===== E
=====
=====  CHILDREN, DO NOT LAUGH
===== URVEY THE TALL GIRAFFE.
      ....+....2....+....3....+....4....+....5....+....6....+....7....+....>...
===== Y SPORTING TO ATTACK
===== AT CANNOT ANSWER BACK.
===== RUMPET FOR A THROAT,
=====  BLOW A SINGLE NOTE.
===== HAT HIS VOICE HE HOARDS;
===== ANY VOCAL CORDS.
=====  HIM, AND FOR HIS WIFE,
===== GIRAFTER LIFE.
===== * * * End of File * * *
====>
                                                      X E D I T  1 File
```

Figure 12. The RIGHT Subcommand — Before and After

SAVE

Use the SAVE subcommand to write the edited file on disk or in an SFS directory without returning control to the environment that invoked the editor. You may also use the SAVE subcommand to change the file identifier.

Format

SAVE	$\left[\begin{matrix} fn \\ = \end{matrix} \left[\begin{matrix} ft \\ = \end{matrix} \left[\begin{matrix} fm \\ = \end{matrix}\right]\right]\right]$

Operands

fn
: indicates the file name of the file to be saved (or membername when editing a member of a MACLIB). If you specify only fn, the file type and file mode remain the same.

ft
: indicates the file type of the file to be saved.

fm
: indicates the file mode of the file to be saved.

Usage Notes

1. If the file mode indicates an accessed minidisk, the minidisk must be accessed read/write. If the file mode indicates an accessed SFS directory, it can be accessed as either read/only or read/write.
2. You must have write authority to an existing file to save it in another user's SFS directory.
3. If you have write authority to another user's SFS directory, you can create a new file in that directory even though you have the directory accessed read/only. The owner of the directory is now the owner of the new file, but because you created the file, you have write authority for the file.
4. If you specify a new file identifier, the original copy of the file on disk or directory is not changed. Changing the file identifier simply creates another copy of the file with a new file identifier.
5. If you try to SAVE a shared file that has been modified by another user during your editing session, the editor stops the SAVE operation and displays the following message:

   ```
   594E File fn ft fm already exists or changed; use FFILE or SSAVE
   ```

   At this point you can decide whether to preserve the other user's changes or write over them. If you want to write over the other user's changes, causing them to be lost, enter SSAVE (abbreviated as SS). If not, change the file identifier so it is unique and reissue the SAVE subcommand to save your changes under a different name. Later, you can determine what the differences are between the two versions of the file and combine them.

To avoid this situation, use the LOCK option on the XEDIT command to prevent other users from changing the file while you are editing it. The LOCK option is the default.

Message 594E is also displayed if you change the file identifier so that it is identical to that of an *existing* file and try to save it.

This "protected SAVE" works in the following way. SAVE is defined as a synonym to PSAVE, which is the protected equivalent of SAVE. SSAVE is a synonym for COMMAND SAVE.

6. To write a file to disk or directory and end the editing session, use the FILE subcommand instead of the SAVE subcommand.
7. If you want a file to be saved automatically at regular intervals, use the SET AUTOSAVE subcommand.
8. The operand fn, ft, or fm may be specified as an equal (=) sign, in which case the corresponding value of the file currently being edited is used.
9. If the automatic save function is in effect (SET AUTOSAVE subcommand), the corresponding AUTOSAVE file is erased when a SAVE is executed.
10. Changing a membername

    If you change the membername (to a unique membername) and the member has previously been written to disk or directory, that copy with the original name of the member is not altered.

    However, if you change the membername while editing a member so that the new membername is identical to that of an existing member, the editor stops the SAVE operation and displays the following warning:

    ```
    594E File fn ft fm already exists or changed; use FFILE or SSAVE
    ```

    If you want to write over the existing member, enter SSAVE (abbreviated as SS). If not, change the membername so that it is unique and reissue the SAVE subcommand.
11. Members of a MACLIB must have a file type of MEMBER.
12. Members with identical names contained in libraries with identical names, but residing on different disks or directories, will not be written over if you attempt to change the file mode. The following warning will be issued.

    ```
    594E File fn ft fm already exists or changed; use FFILE or SSAVE
    ```

    For example:

    ```
    PROJA MACLIB A1               PROJA MACLIB B1
    ENTER                         ENTER
    LEAVE                         LEAVE
    FORMAT                        FORMAT
    COMPUTE                       COMPUTE
    ```

    If you are editing the ENTER member in the PROJA MACLIB that resides on your file mode B and attempt to change the file mode by issuing "SAVE = = A," the warning message is issued.

Messages

002E	File *fn ft fm* not found [RC=28]
007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
033E	File *fn ft fm* is not a library [RC=32]
037E	Filemode *mode* is accessed as read/only [RC=12]
039E	No entries in library *fn ft fm* [RC=32]
048E	Invalid mode *mode* [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
069E	Filemode *mode* not accessed [RC=36]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, 99, or 100]
105S	Error *nn* writing file *fn ft fm* on disk [RC=55, 70, 76, 99, or 100]
157S	MACLIB limit exceeded [RC=88]
167S	Previous MACLIB function not finished [RC=88]
229E	Unsupported OS dataset, error *nn* [RC=80, 81, 82, 83, or 84]
520E	Invalid operand: *operand* [RC=5]
531E	Disk or file space is full; set new filemode or clear some space [RC=13]
532E	Disk or file space is full; AUTOSAVE failed
554E	Not enough virtual storage available [RC=104]
559W	Warning: empty file not written to disk
560E	Not enough space for serialization between TRUNC and LRECL
579E	Records truncated to *nn* when added to *fn ft fm* [RC=3]
594E	File *fn ft fm* already exists or changed; use FFILE or SSAVE [RC=3]
598S	Unable to build update file: internal list destroyed [RC=7]
599S	Unable to build update file: serialization destroyed [RC=7]
622E	Insufficient free storage for reading map [RC=104]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1141W	User filespace threshold exceeded
1215E	File *fn ft fm* is locked by another user [RC=70]
1258E	Not authorized to write file *fn ft fm* [RC=12]
1262S	Error *nn* opening file *fn ft fm* [RC=55, 70, 76, 99, or 100]
1301S	Rollback error *nn*, file *fn ft fm* left open

Return Codes

0	Normal
3	Truncated or spilled, or file already exists
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
7	Error building the update file
12	Minidisk defined in file mode is read-only; or not authorized to write to file
13	Minidisk or file space is full
20	Invalid character in file name or file type
24	Invalid file mode
28	File not found
31	A rollback occurred
32	File is not a library, or library has no entries, or file is not fixed, 80 char. records
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error

80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
84	Unsupported OS data set
88	Previous Maclib function not finished, or Maclib limit exceeded
99	A required system resource is not available
100	Error reading|writing file from|to disk
104	Insufficient virtual storage

SCHANGE (Macro)

Use the SCHANGE (selective change) macro to locate every occurrence of a string and to change that string only when you choose.

The SCHANGE macro can be used when it has been assigned to a PF key (via the SET PF*n* subcommand). You can also type a CLOCATE or CHANGE subcommand on the command line before pressing the PF key assigned to the macro SCHANGE. By default, the SCHANGE macro automatically repeats the CLOCATE or CHANGE subcommand saved in the the LASTLORC buffer (see SET LASTLORC subcommand in this book.)

If used with CLOCATE, only one PF key is needed. If used with CHANGE, two PF keys are required: one to locate the string; the other to perform the change.

Each time the PF key assigned to SCHANGE (via the SET PFn subcommand) is pressed, the cursor is placed under the next occurrence of the string specified in the CHANGE or CLOCATE subcommand. The search for the string starts with the current column in the current line and continues throughout the file.

If a CHANGE /string1/string2/ subcommand has been typed in the command line, pressing the *second* PF key will change string1 to string2. (This second PF key has no effect when the CLOCATE subcommand is used.)

For an example of how to use the SCHANGE macro when it has been assigned to a PF key, see the *VM/SP System Product Editor User's Guide.*

Format

SCHANGE	[*keynumber*]

Operand

keynumber
is the PF key number that causes the CHANGE to be executed, after the string has been located by pressing the PF key assigned to SCHANGE. SET PF5 SCHANGE 6 is the default assigned by the editor, but any numbers can be used.

Usage Notes

1. Pressing the PF key assigned to SCHANGE causes the cursor to move but does not cause the line pointer to move (unless the screen is scrolled forward). The column pointer moves to the column where the specified string begins.
2. The screen scrolls forward automatically after all occurrences of the string have been located on the current screen.

   The screen does not scroll forward if the string does not appear in the rest of the file.
3. Pressing the PF key assigned to SCHANGE places the cursor under the first character of the string.

   Pressing the second PF key causes string1 to be replaced by string2.

4. The advantage of using SCHANGE with CLOCATE (rather than CLOCATE and =) is that the cursor is placed under the matching string.
5. SCHANGE will not change a string located to the left or right of the screen.
6. SCHANGE can be assigned to a PA key instead of a PF key, however, the key number specified on the SCHANGE subcommand must be that of a PF key.
7. When AUTOSAVE is set ON, the file is not autosaved until SCHANGE completes execution. Entering a subcommand other than the two PF keys assigned to SCHANGE and CHANGE completes execution of the SCHANGE macro.

Responses

Each time a string is located, the cursor is placed under the first character of the string, and the line containing the string is highlighted.

If a CLOCATE subcommand is typed in the command line or saved in the LASTLORC buffer and the first PF key is pressed, the following message is displayed when a string is located:

```
Target string1 found
```

If a CHANGE subcommand is typed in the command line or saved in the LASTLORC buffer and the first PF key is pressed, the following message is displayed when a string is located:

```
String string1 found; --- PFnn set for selective CHANGE
```

When the second PF key is pressed, the following message is displayed:

```
String string1 changed to string2
```

If you press the PF key assigned to SCHANGE without first typing a CHANGE or CLOCATE in the command line, and the LASTLORC buffer does not contain a CHANGE or CLOCATE subcommand, the following message is displayed:

```
No CHANGE or CLOCATE subcommand specified
```

Messages

520E	Invalid operand: *operand* [RC = 5]
525E	Invalid {PFkey|PFkey/PAkey} number [RC = 5]
529E	Subcommand is only valid in {display|editing} mode [RC = 3]
545E	Missing operand(s) [RC = 5]
561E	Cursor is not on a valid data field [RC = 1 or 3]
569E	No CHANGE or CLOCATE subcommand specified [RC = 5]
574E	Change not valid {with CLOCATE after cursor movement}
586E	Not found [RC = 2]

Return Codes

0	Normal
2	Not found
3	Invalid placement of cursor, or subcommand is valid only for display terminal
5	Invalid or missing operands
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro caller from the last file in the ring

SET

Use the SET subcommand to change the settings of various editing options while editing is in progress. Only one option may be specified in each SET subcommand. SET cannot be issued without an option. Use the QUERY subcommand to display the initial settings of SET options or the values set by previously issued SET subcommands.

The options available with SET are summarized in the following list. A complete description of each option follows.

ALT	**IMPcmscp**	**SCReen**
APL	**LASTLorc**	**SELect**
ARBchar	**LINENd**	**SERial**
AUtosave	**LRecl**	**SHADow**
BRKkey	**MACRO**	**SIDcode**
CASE	**MASK**	**SPAN**
CMDline	**MSGLine**	**SPILL**
COLOR	**MSGMode**	**STAY**
COLPtr	**NONDisp**	**STReam**
CTLchar	**NULls**	**SYNonym**
CURLine	**NUMber**	**TABLine**
DISPlay	**PAn**	**TABS**
ENTer	**PACK**	**TERMinal**
ESCape	**PENDing**	**TEXT**
ETARBCH	**PFn**	**TOFEOF**
ETMODE	**Point**	**TRANSLat**
FILler	**PREfix**	**TRunc**
FMode	**RANge**	**VARblank**
FName	**RECFm**	**Verify**
FType	**REMOte**	**WRap**
FULLread	**RESERved**	**Zone**
HEX	**SCALe**	**=**
IMage	**SCOPE**	

Notes:

1. The subcommand name SET is generally optional.
2. The editor assigns initial settings for each SET subcommand option. When you issue a SET subcommand, only those operands that you override are changed. All other operands retain their initial setting or default value.

SET ALT

Use the ALT option to change the number of alterations that have been made to the file since the last AUTOSAVE and/or since the last SAVE. This count is displayed in the file identification line as Alt=n.

Format

SET	**ALT** *n* [*p*]

Operands

n
: represents the alteration count since the last AUTOSAVE.

p
: represents the total alteration count since the last SAVE. If you omit p, the total alteration count since SAVE remains unchanged.

Initial Setting

Both counts are initially set to zero.

Usage Notes

1. Both counts may be reset. If only one operand is specified (SET ALT n), the first count is reset. If two operands are specified, then both counts are reset.
2. The subcommand "SET" is required with this option (to avoid conflict with the ALTER subcommand).
3. The alteration count represents the number of subcommands that were executed which changed the file and the number of lines modified when typing on the full screen. The alteration count will be incremented whenever any action is taken to modify a line, even if the line has not actually been changed from its original state. For example, this will occur when a character in a line is overtyped or if the ERASE EOF key is pressed when the cursor is in column one.

   Macros that issue numerous commands that change the file may increase the count by more than one.
4. SET ALT is intended to be used from within a macro. It should not be used for interactive editing because it can interfere with the AUTOSAVE function.

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET APL

Use the APL option to inform the editor and CMS if APL keys are available on the terminal.

Format

[SET]	APL ON OFF

Operands

ON
specifies that APL keys are available. You must issue a SET APL ON before using these keys so that proper character code conversion takes place.

OFF
specifies that no code conversion is to be performed for APL keys.

Initial Setting

`Defined by the CMS SET APL setting.`

Usage Notes

1. If a terminal is equipped with the APL feature, special APL keys are available. Before using these keys, you must inform the editor so that proper character code conversion takes place. There are two ways to inform the editor:
   a. Issue CMS SET APL ON.
   b. Issue the editor SET APL ON subcommand.
2. Because the conversion is costly, it is recommended that you issue the XEDIT subcommand SET APL OFF when you stop using the special keys.
3. Issuing SET APL ON while TEXT is ON causes TEXT to be set to OFF. Similarly, issuing SET TEXT ON while APL is ON causes APL to be set to OFF.
4. Changing the APL setting for XEDIT also changes the APL setting for CMS.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET ARBCHAR

Use the ARBCHAR option to define an arbitrary character to be used in a target definition. An arbitrary character used between character strings in a target definition means, "all intervening characters are to be ignored."

Format

[SET]	**ARBchar**	**ON** **OFF**	[*char*]

Operands

ON
turns on the use of a special character as an arbitrary character.

char
is the special character. The initial setting is a dollar sign ($). Several consecutive occurrences of the arbitrary character are equivalent to one.

OFF
turns off the use of a special character as an arbitrary character without changing the current character definition.

Initial Setting

```
ARBCHAR OFF $
```

Examples

1. `locate  /air$plane/`

   will locate in the file

   `"the airplane was landing"`

   and will also locate in the file

   `"cold air surrounded the plane"`

2. `clocate  /($)/`

   will locate the first expression in the line that is in parentheses.

3. Any special character can be SET as an arbitrary character. If the arbitrary character is used consecutively a number of times in a target definition, it is treated as one, thus allowing, for example, the use of ellipsis.

   For example:

   ```
   set arbchar on .
   locate  /air...plane/
   ```

   is equivalent to

   ```
   locate  /air.plane/
   ```

Both will locate

`"airplane"` or `"cold air surrounded the plane"`

in the file.

4. Using CHANGE with SET ARBCHAR:

   String2 must not contain more arbitrary characters than string1. If it does, the following error message is displayed:

   ```
   String2 contains more arbitrary characters than string1.
   ```

   For example:

```
Subcommand:  set arbchar on $
File Line:   the white house with several large windows


Subcommand:  c /the$house$windows/a$farmhouse$two$shutters/
Result:
```

```
511E    String2 contains more arbitrary characters than string1
```

```
Subcommand:  c /the$house$windows/a$farmhouse$shutters/
Result:      a white farmhouse with several large shutters
```

String2 can have fewer arbitrary characters than string1. For example:

```
File Line:   the white house with several large windows
Subcommand:  c/the$house$windows/a$large farmhouse/
Result:      a white large farmhouse
```

Special use of an arbitrary character:

If a dollar sign ($) is the arbitrary character, it may be used in string1 and/or string2 in the following ways:

string1: `/$A/`

The $ means, "the string starting in the zone1 column and ending with the character that precedes A." (If no zone is defined, the string starts in column 1.)

Example:

```
change /$A/A/
```

deletes all characters that precede A within the same zone.

string1: `/B$/`

The $ means, "the string starting with the character following **B** and ending at the truncation column." If SET ZONE has been issued, the string ends at the zone2 column rather than the truncation column.

Example:

```
change /B$/B/
```

deletes the characters that follow **B**.

Messages

511E	String2 contains more arbitrary characters than string1 [RC = 5]
520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET AUTOSAVE

Use the AUTOSAVE option to set or reset the automatic save function of the editor. When the automatic save function is in effect, the editor automatically issues a SAVE subcommand each time the specified number of alterations is made.

Format

[SET]	**AUtosave** *n* [*mode* / **A̲**] **OFF**

Operands

n
is a number that specifies the frequency of the automatic save function. One SAVE subcommand is issued for every n subcommands that change, delete, or add lines to the file. In a full screen environment, each line changed (by over-typing) counts as one. The automatic SAVE occurs when this number equals or exceeds n.

OFF
turns off the automatic save function.

mode
specifies the accessed minidisk or SFS directory on which the file is written. The default mode is A.

Initial Setting:

```
AUTOSAVE  OFF
```

Usage Notes

1. You can use the QUERY AUTOSAVE subcommand to display the current AUTOSAVE setting. The file identification line indicates the number of changes since the last AUTOSAVE (Alt=n).
2. When AUTOSAVE is in effect, SAVE or FILE subcommands will erase the corresponding AUTOSAVE file. The QUIT subcommand will not erase it.
3. If the system crashes during an editing session, you can recover all changes made up to the time of the last automatic save. To do this, replace the original file with the AUTOSAVE file using the CMS COPYFILE command with the REPLACE option. Then, erase the AUTOSAVE file and resume editing. COPYFILE will keep any SFS authorities and aliases associated with the original file.

   Another way to recover your changes after a system crash is to erase the original file and rename the AUTOSAVE file to the original file; however, this method will cause you to lose any SFS authorities and aliases associated with the original file.
4. To recover from an erroneous change to the file, for example, a bad global change, you should immediately issue a QUIT subcommand, replace the original file with the AUTOSAVE file by using the REPLACE option of the CMS

COPYFILE command, erase the autosave file, and continue. However, if the ALT count in the file identification line is 0, the erroneous change has already been saved and this technique will not recover from it.

5. While an XEDIT macro is executing, the automatic save won't be performed. The autosave will be done when the macro completes execution.
6. The automatic save function requires that the file mode be accessed as read/write. Thus, even though you may XEDIT files in another user's directory, the AUTOSAVE file must be targeted for one of your own directories or minidisks.

Notes for Macro Writers

You can use the EXTRACT/AUTOSAVE/ subcommand to return the autosave information for further processing by a macro.

Responses

Each time an automatic save occurs, the editor writes the file to your file mode A (or the mode specified in the SET AUTOSAVE subcommand). It assigns the file a unique numerical file name and a file type of AUTOSAVE. QUERY UNIQUEID displays the unique AUTOSAVE file name.

The identifier has the form *rrrnnnnn* where *rrr* is the number of times XEDIT was called recursively and *nnnnn* is the current autosave number. Note that when the recursion level, *rrr*, is less than 100, the leading zero(es) will be dropped. Consequently, the file name can have as many as eight digits or as few as six, depending on the recursion level.

If no other AUTOSAVE file exists, the file name and file type are:

```
100001  AUTOSAVE
```

In this example, the left-most 1 is the recursion level and the right-most 1 is the current autosave number.

If an AUTOSAVE file exists, the file name is the next available number.

When an autosave occurs, the following message is displayed:

```
510I Autosaved as fn ft fm
```

Messages

037E	Filemode *mode* is accessed as read/only [RC = 12]
048E	Invalid mode *mode* [RC = 24]
069E	Filemode *mode* not accessed [RC = 36]
520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Mode is read only
24	Invalid file mode
36	Corresponding mode is not accessed

SET BRKKEY

Use the SET BRKKEY option to inform the editor whether CP should break in when the "BRKKEY" (defined by CP TERMINAL BRKKEY) is pressed.

Format

[SET]	**BRKkey** **ON** *key* **OFF**

Operands

ON
indicates that pressing the key defined as the BRKKEY causes a control break-in by CP. If you set BRKKEY ON without assigning it to a specific key, then XEDIT sets BRKKEY ON PA1 by default.

OFF
indicates that no control break-in is desired and that XEDIT can use the definition of the key that was depressed. You can then use this key in XEDIT as it has been defined by the SET PFn or SET PAn subcommand.

key
is the PA1 key or any PF key (1-24). PA1 is the default. Be sure that the key you specify is available on the terminal you are using.

Initial Setting

```
Defined by the CP BRKKEY setting.
```

Usage Notes

1. The BRKKEY is initially defined by CP as the PA1 key. To define a BRKKEY, see the TERMINAL command in the *VM/SP CP General User Command Reference*.
2. Issuing the editor SET BRKKEY subcommand changes the CP TERMINAL BRKKEY setting. (The new setting remains in effect after you leave the XEDIT session).
3. If the BRKKEY is set to ON and you enter full-screen CMS, the BRKKEY setting is changed to OFF.

Messages

520E	Invalid operand: *operand* [RC=5]
525E	Invalid PF/PA key number [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET CASE

Use the CASE option to control how letters are entered into the file, to control how commands are translated from the command line (whether the commands are entered from the terminal or from the console stack), and to specify if uppercase and lowercase letters are to be significant in target searches.

Format

[SET]	CASE	Uppercase Mixed	[Respect Ignore]

Operands

Uppercase
indicates that the editor is to translate all lowercase letters to uppercase (whether the lines are entered from the terminal or from the console stack).

Mixed
indicates that the editor is not to translate uppercase and lowercase letters (whether the lines are entered from the terminal or console stack).

Respect
In target searches, an uppercase letter will not match a lowercase letter (and vice versa). For example:

```
/This Text/
```

will not locate

```
"this text"
```

in the file.

Ignore
In target searches, uppercase and lowercase representations of the same letter will match. For example:

```
locate /THIS TEXT/
```

will locate

```
"this text"
```

in the file.

Initial Setting

Based on file type. See "Appendix A."

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET CMDLINE

Use the CMDLINE option to specify the position of the command line on the screen.

Format

[SET]	**CMDline**	**On** **OFf** **Top** **Bottom**

Operands

On
defines the last two lines of the screen as the command input lines.

OFf
removes the command line from the screen.

Top
defines the second line of the screen as the command line.

Bottom
defines the last line on the screen as the command input line.

Initial Setting

```
CMDLINE  ON
```

Usage Notes

1. When CMDLINE is set to TOP, BOTTOM, or OFF, no XEDIT status area is displayed.
2. Before setting CMDLINE OFF, you should establish a method of resetting the CMDLINE, perhaps by setting a PF key or through the use of a prefix macro, if you wish to use the command line again.
3. The command line cannot be overlayed by a reserved line unless it is first set OFF.
4. If CMDLINE ON is in effect and you split the screen vertically (see SET SCREEN), only one line is available as a command line when two views are adjacent. However, when you return to a single screen, both lines are again available. SET CMDLINE ON is invalid when issued from a vertical split screen.
5. With CMDLINE TOP and the default SET MSGLINE setting (line 2), a message overlays the command line, including the arrow. You must press the CLEAR key (or any key that does not produce a message) to restore the command line. To avoid this situation, assign the message line to line 1 or line 3 (see SET MSGLINE) when using CMDLINE TOP.
6. When SET CMDLINE is entered with the ON, TOP, or BOTTOM operands, the cursor will be positioned in the first column of the command line.

Messages

520E	Invalid operand: *operand* [RC=5]
526E	Option *option* valid in display mode only [RC=3]
545E	Missing operand(s) [RC=5]
568E	Subcommand not valid with this screen definition [RC=5]

Return Codes

0	Normal
3	Operand is valid only for display terminal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET COLOR

Use the COLOR option to associate specific colors with certain areas of the XEDIT screen.

Format

```
[SET]   COLOR  field  [color][exthi] [High|Nohigh] [PSs]
               *
```

Operands

*
indicates all of the fields listed below.

field
can be any of the following areas on the screen.

Arrow
the arrow pointing to command line.

Cmdline
the line where commands are entered.

CUrline
the file line that is the current line.

Filearea
file data area, excluding the current line, the TOF and EOF lines.

Idline
the file identification line on line 1 of the screen.

Msgline
the area used to display messages.

Pending
the pending macro or subcommand as entered in the prefix area.

PRefix
the five-position area that lies either to the left or the right of each line of the file, as defined by SET PREFIX.

Scale
the scale line.

SHadow
shadow line(s) resulting from selective line editing (See Appendix B.)

STatarea
status area in lower right-hand corner of screen which displays the current status of your editing session.

Tabline
the line which displays a "T" in every tab column, according to the current tab settings.

TOfeof
the top of file and end of file notices.

color
can be any one of the following:

Blue	Turquoise
Red	Yellow
Pink	White
Green	Default.

If you omit color, the current color is not changed.

exthi
extended highlighting can be any one of the following:

```
BLInk
REVvideo  (reverse video)
Underline
NONe.
```

If you omit exthi, the current extended highlighting setting is not changed.

High
field is to be displayed highlighted

Nohigh
field is to be displayed not highlighted (normal intensity)

If you omit High|Nohigh, the current High|Nohigh setting is not changed. Note that High|Nohigh is affected by usage note 1.

PSs
specifies the programmed symbol set to be used. PS0 indicates that the default character set should be used. PSs can be any of the following:

PS0	PSD
PSA	PSE
PSB	PSF
PSC.	

Initial Settings

FIELD	COLOR	EXTHI	HIGH/NOHIGH	PSs
ARROW	DEFAULT	NONE	HIGH	PS0
CMDLINE	DEFAULT	NONE	NOHIGH	PS0
CURLINE	DEFAULT	NONE	HIGH	PS0
FILEAREA	DEFAULT	NONE	NOHIGH	PS0
IDLINE	DEFAULT	NONE	HIGH	PS0
MSGLINE	RED	NONE	HIGH	PS0
PENDING	DEFAULT	NONE	HIGH	PS0
PREFIX	DEFAULT	NONE	NOHIGH	PS0
SCALE	DEFAULT	NONE	HIGH	PS0
SHADOW	DEFAULT	NONE	NOHIGH	PS0
STATAREA	DEFAULT	NONE	HIGH	PS0
TABLINE	DEFAULT	NONE	HIGH	PS0
TOFEOF	DEFAULT	NONE	NOHIGH	PS0

Usage Notes

1. The SET COLOR subcommand accepts the color, extended highlighting, and programmed symbol set operands regardless of whether or not the device has the ability to use those attributes. However, the action taken is dependent upon the device. For example, HIGH and NOHIGH are ignored on a 3279 display (unless the color was DEFAULT), color is ignored on a 3278 display, and color, extended highlighting, and character set are ignored on a 3277 display. Also, QUERY and EXTRACT return all the current settings, even those that may be ignored on any given terminal.
2. The attributes following the field keyword can appear in any order. Only one attribute is required following the field.
3. If TOFEOF is set off (see SET TOFEOF in this publication), the extended highlighting attributes (reverse video and underline) still will be visible.
4. The color and extended highlighting of the current line override the color and extended highlighting of the filearea and TOFEOF for any line that is the current line.
5. The color and extended highlighting of the scale override the color and extended highlighting of the tabline when the scale and tabline are on the same line.
6. On 3270 terminals equipped with the Programmed Symbol (PS) feature you can specify alternate character sets to be used on your terminal. The characters in these sets use different symbols, such as a different style or font, than the default character set.

   New character sets are loaded using the Graphic Data Display Manager (GDDM) program product (5748-XXH) or an application that loads programmed symbol sets. During the loading of programmed symbol sets, symbol set identifiers (SSIDs) are specified by the user. In XEDIT, the allowable SSIDs are in the decimal range 193 through 198. These SSIDs correspond to PSA through PSF, respectively (that is, 193 to PSA). Character sets should be loaded before invoking XEDIT. If a new set is loaded, it will not be recognized and the last character set loaded prior to invoking the editor will be displayed. Likewise, if you drop a programmed symbol set, the editor is not aware that the set is no longer available and attempts to use that set. This may cause I/O errors when the display is written (resulting in SET TERMINAL TYPEWRITER). To avoid problems, do not load or delete program symbol sets when in the XEDIT environment.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET COLPTR

Use the COLPTR option only with typewriter terminals to control whether or not the column pointer (represented as an underscore character) is displayed.

Format

[SET]	**COLPtr**	**ON** **OFF**

Operands

ON
displays the column pointer.

OFF
removes the column pointer from the display.

Initial Setting

`COLPTR ON` (typewriter terminals only)

Usage Note

In order for the column pointer to be displayed, your typewriter terminal must be equipped with a backspace key.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET CTLCHAR

Use the CTLCHAR option to define a control character, which specifies that part of a reserved line is to be displayed with or without the following features:

- Color
- Extended highlighting
- Highlighting
- Programmed symbol set
- Protection
- Visibility.

This option is designed to be issued from a macro, particularly when you are creating display panels and other interactive information. It applies to lines reserved by the SET RESERVED subcommand.

Format

```
SET       CTLchar char  Escape
                        OFF
                        Protect   [color] [exthi] [High |Nohigh |Invisible] [PSs]
                        Noprotect [color] [exthi] [High |Nohigh |Invisible] [PSs]
                  OFF
```

Operands

char
: is any character, uppercase or lowercase, which is interpreted as a control character when embedded in the text to be displayed. If specified with the ESCAPE operand, this character identifies the next character in the text as a control character. No more than 63 characters can be defined.

OFF
: resets all control characters that have been defined (SET CTLCHAR OFF), or resets a specified character (SET CTLCHAR char OFF).

Escape
: specifies that the designated character (char) is a control character identifier; that is, when this character appears in the text, the next character in the text is interpreted as a control character.

Protect
: specifies that the string following "char" is to be displayed in a protected field.

Noprotect
: specifies that the string following "char" is to be displayed in an unprotected field.

color
: color can be any one of the following:

```
Blue     Turquoise
Red      Yellow
Pink     White
Green    Default.
```

If you omit color, the default is DEFAULT (the default base color).

exthi
extended highlighting can be any one of the following:

```
BLInk
REVvideo - reverse video
Underline
NONe.
```

If you omit exthi, the default is NONE (no extended highlighting).

High
specifies that the string following "char" is to be displayed highlighted. If you omit High|Nohigh|Invisible, Nohigh is assigned.

Nohigh
specifies that the string following "char" is to be displayed not highlighted (normal intensity). If you omit High|Nohigh|Invisible, Nohigh is assigned.

Invisible
defines a field where the characters are not displayed, for example, a password. If you omit High|Nohigh|Invisible, Nohigh is assigned.

PSs
the programmed symbol set can be any of the following:

```
PS0     PSD
PSA     PSE
PSB     PSF
PSC.
```

PS0 indicates that the default character set will be used.

Initial Setting

No control characters are defined. CTLCHAR is set OFF.

Notes for Macro Writers

1. Using control characters to specify the display within a reserved line requires three steps:

   - Define the control character identifier using the ESCAPE operand of SET CTLCHAR,
   - Define the control character with its associated features,
   - Use the SET RESERVED subcommand with the control character identifier and control character to specify the display within the reserved line.

   See the following "Examples" section for an example of each of these steps.

2. If you use SET CTLCHAR to set up multiple fields within reserved lines, particularly input (unprotected) fields, you should use READ with the TAG option. Multiple fields on a single reserved line are stacked separately, and a tag identifying the origin of each line is inserted. For more information on using READ with the TAG option, see the READ subcommand.

3. The SET CTLCHAR subcommand accepts the color, extended highlighting and programmed symbol set operands whether or not the terminal has the ability to use those attributes. However, the action taken depends upon the device. For example, HIGH and NOHIGH are ignored on a 3279 display (unless the color was DEFAULT), color is ignored on a 3278 display, and color, extended highlighting, and character set are ignored on a 3277 display. Also, QUERY

and EXTRACT return all the settings, even those that may be ignored on any given terminal.

4. On 3270 terminals equipped with the Programmed Symbol (PS) feature you can specify alternate character sets to be used on your terminal. The characters in these sets use different symbols, such as a different style or font, than the default character set.

   New character sets are loaded using the Graphic Data Display Manager (GDDM) program product (5748-XXH) or an application that loads programmed symbol sets. Character sets should be loaded before invoking XEDIT. If a new set is loaded, it will not be recognized and the last character set loaded prior to invoking the editor will be displayed. Likewise, if you drop a programmed symbol set, the editor is not aware that the set is no longer available and attempts to use that set. This may cause I/O errors when the display is written (resulting in SET TERMINAL TYPEWRITER). To avoid problems, do not load or delete program symbol sets when in the XEDIT environment.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
695E	Cannot define more than 63 CTLCHARs [RC=4]

Return Codes

0	Normal
4	Too many control characters defined
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

The following subcommands are issued:

`set ctlchar ! escape`

Defines ! as a control character identifier. When ! appears in the text, the next character is interpreted as a control character.

`set ctlchar % protect high`

When % appears in the text, preceded by the control character identifier (!), the data that follows it is displayed protected and highlighted.

`set ctlchar " protect nohigh`

When " appears in the text, preceded by the control character identifier (!), the data that follows it is displayed protected and not highlighted.

`set ctlchar ? protect blue underline psa`

When ? appears in the text, preceded by the control character identifier (!), the data that follows it is protected, underlined, and displayed in blue using programmed symbol set A.

`set reserved 3 noh This is an !%example!" of selective !?highlighting!"`

When the screen is displayed, the word "example" is highlighted on line 3 and the word "highlighting" is protected, underlined, and displayed in blue using programmed symbol set A (PSA).

SET CURLINE

Use the CURLINE option to define the position of the current line on the screen.

Format

[SET]	**CURLine ON** M[+*n* |-*n*] | [±|-]*n*

Operands

ON
displays the current line on the screen.

M[+*n*|-*n*]
specifies the line in which the current line is displayed. M stands for "middle of the screen" (rounded up for odd sized screens). M may be combined with a constant positive or negative integer to mean n lines below the middle of the screen (M + n), or n lines above the middle of the screen (M − n). For example, CURLINE ON M means the "middle of the screen," and CURLINE ON M+3 means " three lines below the middle of the screen."

[±|-]*n*
specifies the line in which the current line is displayed. n or +n (+ is implicit) specifies that the current line is displayed n lines from the top of screen; −n specifies that the current line is displayed n lines from the bottom of screen. For example, CURLINE ON −3 means that the current line is three lines from the bottom of the screen.

Initial Setting

`CURLINE ON M` (middle of the screen)

Usage Note

1. Under certain conditions, the current line as it appears on your display screen does not have the same line number that you specified on the SET CURLINE subcommand. This occurs when the CURLINE line setting is the same as the line setting for any of these SET options:

   - SET RESERVED
   - SET SCALE
   - SET TABLINE
   - SET MSGLINE

   When the CURLINE line setting is the same as the line setting for SET SCALE, for example, the scale occupies the specified line and the current line is moved down to the next available line.

   Note: Other subcommands that refer to CURLINE (EXTRACT, for example) use the true current line regardless of where it happens to appear on the display screen.

Messages

520E	Invalid operand: *operand* [RC = 5]
521E	Invalid line number [RC = 5]
526E	Option *option* valid in display mode only [RC = 3]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
3	Operand is valid only for display terminal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET DISPLAY

Use the DISPLAY option to specify which selection levels of lines (as defined by SET SELECT) are displayed. For an illustration of how SET DISPLAY can be used effectively in combination with the other selective line editing subcommands (SET SELECT, SET SCOPE, and SET SHADOW), see SET SELECT in this publication.

Format

[SET]	**DISPlay** *n* **1** [*n* **2**| *****]

Operands

n1
: is a positive whole number that represents a selection level of lines that you wish to display. If n1 is specified alone, only the lines having a selection level of n1 are included in the display.

n2|*
: is a positive whole number that represents a selection level of lines. When both n1 and n2 are specified, all lines with selection levels between n1 and n2 inclusive are included in the display. The value of n2 must be equal to or greater than that of n1. If n2 is specified as an asterisk (*), all lines with selection levels starting with n1 and greater are displayed.

Initial Setting

```
DISPLAY 0 0
```

Usage Notes

1. To display the entire file, use SET DISPLAY 0 *. If you use "SET DISPLAY 0 *," you cannot subsequently use the X prefix macro to exclude lines. You must first reset the DISPLAY level to something other than asterisk (*).
2. For examples of this and other selective line editing subcommands, you can examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display. Appendix B, "Effects of Selective Line Editing Subcommands" on page 405.

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
600E	First selection level (*nn*) cannot be greater than second selection level (*nn*) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET ENTER

Use the ENTER option to define a meaning for the keyboard ENTER key or to remove the meaning associated with the ENTER key.

Format

[SET]	**ENTer**	<u>**BEFORE**</u> **AFTER** **ONLY** **IGNORE**	*string* **NULLKEY** **COPYKEY** **TABKEY**

Operands

<u>**BEFORE**</u>
specifies that the key definition is executed before the contents of the command line. BEFORE is the default for SET ENTER, unless the string begins with a "?" or "=." For these two strings, ONLY is the default.

AFTER
specifies that the key definition is executed after the contents of the command line.

ONLY
specifies that only the key definition is executed, thereby ignoring the command line.

IGNORE
specifies that the key definition is ignored when something is entered on the command line, thereby only executing the command line.

string
is any XEDIT subcommand or macro (including CP, CMS, or EXEC), which is executed when the ENTER key is pressed. If string is null, the ENTER key will be undefined.

NULLKEY
When the ENTER key is pressed, blank characters are changed to nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, then blanks will be changed to nulls on the file line with which that prefix area is associated.

TABKEY
When the ENTER key is pressed, the cursor is moved into the next tab column, as defined by the SET TABS subcommand.

COPYKEY
When the ENTER key is pressed, the exact content of the virtual screen is copied into the printer spool.

Initial Setting

```
ENTER IGNORE COMMAND CURSOR CMDLINE 1 PRIORITY 30
```

Usage Notes

1. When you use the ENTER key set to COPYKEY, you should close the printer at the end of the editing session. If your printer is defined as a 3203, 3211, 3262, 4245, or 3289e, see the *VM/SP CP General User Command Reference* under LOADVFCB for information on the default FCB.
2. When you press the ENTER key set to TABKEY, COPYKEY, or NULLKEY, no screen changes are processed, including the command line, and the keywords BEFORE, AFTER, ONLY, and IGNORE have no effect. A PA/PF key would have to be used to process the command line and/or to change the ENTER key definition.
3. ONLY as part of the ENTER key setting should be used with extreme care, since the command line will not be processed. A PA/PF key would have to be used to process the command line and/or to change the ENTER key definition.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
622E	Insufficient [free] storage [*for copykey*] [RC=104]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
104	No storage is available

Example

Setting the priority to 5 as shown below will prevent the cursor from moving to the command line if you are in the file area when the ENTER key is pressed.

```
set enter ignore command cursor cmdline 1 priority 5
```

SET ESCAPE

Use the ESCAPE option on a typewriter terminal to allow you to enter a subcommand when you are in input mode, without leaving input mode.

Format

[SET]	**ESCape** **ON** [*char*] **OFF**

Operands

ON
turns on the use of an escape character.

char
specifies a character to be used.

OFF
turns off the use of an escape character.

Initial Setting

`ESCAPE  ON` (typewriter terminal)
`ESCAPE OFF` (display terminal)

A default escape character is assigned according to file type; see "Appendix A."

Usage Notes

1. The escape character must appear in column 1. It identifies the line being entered as a subcommand.
2. The default escape character can be displayed by QUERY ESCAPE.
3. This subcommand has no meaning on a display terminal.
4. The escape character can be specified in hexadecimal if SET HEX ON is in effect.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET ETARBCH

Use the ETARBCH option to define an extended arbitrary character within a Double-Byte Character Set (DBCS) string. An extended arbitrary character used between two character strings in a target definition means "all intervening characters are to be ignored when searching for a match in the file."

Format

[SET]	**ETARBCH**	**ON** **OFF**	[*char*]

Operands

ON
turns on the use of a double-byte special character as an extended arbitrary character.

char
specifies the double-byte character to be used as the extended arbitrary character.

OFF
turns off the use of a special character as an extended arbitrary character without changing the current character definition.

Initial Setting

The initial setting is a double-byte monetary symbol (X'425B').

`ETARBCH OFF`	`"]$"`	(X'425B') on a 3278 and 3279 terminal
`ETARBCH OFF`	`""$"`	(X'425B') on a 3277 terminal
`ETARBCH OFF`	¥	(X'425B') on a 5550 Multistation

Usage Notes

1. See Appendix F for information on using DBCS strings in the XEDIT environment.
2. Extended arbitrary characters can be used in targets and in the CHANGE, COUNT, and SPLIT subcommands.
3. The extended arbitrary character may be specified in hexadecimal if SET HEX ON is in effect.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET ETMODE

Use the ETMODE (extended mode) option to specify that XEDIT should recognize Double-Byte Character Set (DBCS) strings. In this mode, XEDIT can manipulate and display DBCS strings on terminals that support Double-Byte Character Sets.

Format

[SET]	**ETMODE** **ON** **OFF**

Operands

ON
specifies that XEDIT will recognize and manipulate DBCS strings that begin with a shift-out character and end with a shift-in character.

OFF
specifies that all data should be treated as 1-byte EBCDIC data.

Initial Setting

The initial setting is based on whether the terminal can display double-byte characters. If it can, the initial setting is ON; if not, the setting is OFF.

Usage Notes

1. If full-screen CMS is off or suspended and you disconnect from a terminal that supports DBCS and reconnect to a terminal that does not support DBCS, then the first time you enter the XEDIT command on the new terminal, ETMODE will be set ON. If you disconnect from a terminal that does not support DBCS and reconnect to a terminal that does, then the first time you enter the XEDIT command on the new terminal, ETMODE will be set OFF.
2. See Appendix F for information on using DBCS strings in the XEDIT environment.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Invalid or missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET FILLER

Use the FILLER option to define a character to be used by the editor when a line is expanded; that is, when tab characters are removed and replaced by an appropriate number of filler characters (see the EXPAND subcommand.) The default filler character is a blank.

Format

[SET]	**FILler** [*char*]

Operand

char
specifies a filler character to be used.

Initial Setting

The "blank" character is defined as the filler.

Usage Note

The filler character can be specified in hexadecimal if SET HEX ON is in effect.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET FMODE

Use the FMODE option to change the file mode of the file being edited.

Format

[SET]	**FMode** *fm*

Operand

fm
is the new file mode. You can specify a file mode letter (A through Z) or a file mode letter and number (0-6). If you specify only a file mode letter, the existing file mode number is used.

Usage Notes

1. The specified file mode is used the next time a FILE or SAVE is entered. If the file being edited had been written to disk or directory before, that copy of the file remains unchanged.
2. If the disk or directory specified by file mode already contains a file with the same file name and file type and you enter FILE or SAVE, the editor displays an error message.
3. If the file mode specified is that of a read-only minidisk, and you enter FILE or SAVE, the editor displays an error message.
4. If you are editing a member of a MACLIB, this command will change the mode of the member.

Messages

048E	Invalid filemode *mode* [RC=24]
069E	Filemode *mode* not accessed [RC=36]
520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
555E	File *fn ft fm* already in storage [RC=4]

Return Codes

0	Normal
4	File already in storage
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
24	Invalid file mode
36	Corresponding mode not accessed

SET FNAME

Use the FNAME option to change the file name of the file being edited.

Format

[SET]	**FName** *fn*

Operand

fn
is the new file name; it can be from one to eight characters long.

Usage Notes

1. The specified file name is used the next time a FILE or SAVE is issued. If the file being edited had been written to disk or directory before, that copy of the file remains unchanged.
2. If a file already exists with the specified file name and the same file type and file mode and you issue FILE/SAVE, the editor displays an error message.
3. If you are editing a member of a MACLIB, this command will change the member name.

Messages

062E	Invalid character in fileid *fn ft fm* [RC=20]
520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
555E	File *fn ft fm* already in storage [RC=4]

Return Codes

0	Normal
4	File already in storage
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
20	Invalid character in the string proposed as file name

SET FTYPE

Use the FTYPE option to change the file type of the file being edited.

Format

[SET]	**FType** *ft*

Operand

ft
is the new file type; it can be from one to eight characters long.

Usage Notes

1. The specified file type is used the next time a FILE or SAVE is issued. If the file being edited had been written to disk or directory before, that copy of the file remains unchanged.
2. If a file already exists with the specified file type and the same file name and file mode and you issue FILE/SAVE, the editor displays an error message.
3. When editing a member of a MACLIB, the file type must remain MEMBER.

Messages

062E	Invalid character in fileid *fn ft fm* [RC=20]
520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
555E	File *fn ft fm* already in storage [RC=4]

Return Codes

0	Normal
4	File already in storage
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
20	Invalid character in the string proposed as file type

SET FULLREAD

Use the FULLREAD option to allow recognition by XEDIT and CMS of 3270 null characters in the middle of screen lines. SET FULLREAD changes the CMS setting of the FULLREAD option.

Format

[SET]	**FULLread**	**ON** **OFF**

Operands

ON
in combination with SET NULLS ON, enables XEDIT and full-screen CMS to recognize nulls in the middle of lines, allowing you to enter tabular or pictorial data without worrying about it being "crushed to the left." NULLS ON allows you to use insert mode.

OFF
inhibits transmission of nulls from the terminal to XEDIT. The SET NULLS subcommand can be used in this situation to control the relationship between nulls and blanks in screen data.

Initial Setting

```
Defined by the CMS setting.
```

Usage Notes

1. When FULLREAD ON is issued, nulls at the end of screen lines that are part of a logical line that occupies more than one physical screen line are dropped. This allows you to delete characters in a screen line and still have the line reconstructed flush together even though multi-line 327X lines do not "wrap" when the character delete key (or the insert mode key) is used.
2. Using FULLREAD ON has performance implications. For local channel attached display controllers with several users taking advantage of the SET FULLREAD usability improvement, there is a potential for a significant increase in system response time. For remote attached display controllers, setting FULLREAD ON will result in a noticeable increase in response time. Additionally, due to increased line traffic, the maximum number of terminals that can be supported on a link may be significantly reduced.

   An alternative to using fullread is the 3274 Entry Assist option. See the *VM/SP Terminal Reference* for more information.
3. Setting FULLREAD ON will prevent you from losing any screen changes when you press a PA key and a message is displayed on a cleared screen.

4. A certain terminal configuration, which imposes several restrictions on your XEDIT session, occurs when going through a VM/Passthru Facility (5749-RC1) (PVM) 327X Emulator link to another VM system. These PVM links can be identified by an 'S' to the immediate left of the nodeid in the PVM selection screen. The following is a list of these restrictions:
   a. The subcommand "SET FULLREAD ON" may not be used.
   b. All PA keys (except for the CP defined TERMINAL BRKKEY) are made non-functional.
5. Changing the editor FULLREAD setting also changes the CMS FULLREAD setting.
6. Use of the (—>) key or any PF key set by the SET PFn TABKEY subcommand creates nulls in the middle of the command input area, regardless of the SET NULLS setting. When FULLREAD is OFF, these nulls will be "crushed" to the left. When FULLREAD is ON, the nulls will be converted to blanks. Because these blanks are treated as user-entered characters, they could affect the column placement of the actual user-entered characters in the file area.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

If you enter the following line with NULLS ON and FULLREAD OFF, using the (—>) key to move the cursor between the columns of data:

```
Vermont     Maine     California     Georgia
```

the data will be "crushed" to the left as follows when you press the ENTER key. Nulls are not recognized in the middle of screen lines when NULLS ON and FULLREAD OFF are both set.

```
VermontMaineCaliforniaGeorgia
```

However, entering the same line with NULLS ON and FULLREAD ON using the (—>) key between the columns of data results in the nulls being handled as blanks as follows:

```
Vermont     Maine     California     Georgia
```

SET HEX

Use the HEX option to allow subcommand string operands and targets to be specified in hexadecimal notation. The editor searches for their EBCDIC equivalent.

Format

[SET]	**HEX** **ON** **OFF**

Operands

ON
allows subcommand string operands and targets to be specified in hexadecimal notation.

For example:

```
locate /X'C1C2C3'/
```

is the same as

```
locate /ABC/
```

OFF
turns off the ability to specify string operands and targets in hexadecimal notation.

Initial Setting

```
HEX OFF
```

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET IMAGE

Use the IMAGE option to determine the way the editor handles tab characters (X'05') and backspace characters (X'16') when a line is entered (from a terminal or the console stack).

Format

[SET]	**IMage** **ON** **OFF** **Canon**

Operands

ON

specifies that an input line is reconstructed into an exact typewriter-like image.

Tab characters are expanded with blank (or filler) characters, according to the current SET TABS settings.

Overstrikes are interpreted as corrections, the backspace acting like a character delete and each column being occupied by the last character typed in. For example:

```
set  tabs  2  5  10  etc.


INPUT LINE:  XB_TYBZ


B represents a backspace character.

T represents a tab character.

_ represents an underscore character.


AFTER PROCESSING:  b_bbZ


b represents a blank
```

OFF

specifies that tab and backspace characters are to be left as they are entered.

Canon

specifies that tabs are left as they are entered, that is, tabs are not expanded to blank (or filler) characters.

Backspace characters may be used to produce compound characters, such as underscored words.

Before a line containing backspace characters is inserted in the file, the characters are rearranged according to canonical order, that is, in collating sequence separated by backspaces. For example,

```
INPUT LINE:  XYZBBB_ _ _


B represents a backspace character

_ represents an underscore character.


AFTER PROCESSING:  _BX_BY_BZ
```

This process simplifies searches through the file for a string, because you don't have to know the exact order in which the characters were entered.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Note

The following subcommands and macros are affected by the IMAGE setting:

- ADD (and A and I prefix subcommands)
- COMPRESS
- EXPAND
- FIND, FINDUP, NFIND, NFINDUP
- INPUT
- OVERLAY
- REPLACE
- SPLIT
- VMFOPT
- All targets
- Typing over a line.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET IMPCMSCP

Use the IMPCMSCP option to control whether subcommands that are not recognized by XEDIT are transmitted implicitly to CMS and later to CP (if necessary) for execution.

Format

[SET]	**IMPcmscp** **ON** **OFF**

Operands

ON
specifies that subcommands not recognized by the editor are sent to CMS and later (if needed) to CP for execution; that is, subcommands unknown to XEDIT are considered to be CMS (or CP) commands.

OFF
specifies that unrecognized subcommands are not sent to CMS and CP.

Initial Setting

```
IMPCMSCP  ON
```

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET LASTLORC

Use the LASTLORC (Last Locate or Change) option within a macro to specify the contents of the LASTLORC buffer.

Format

[SET]	**LASTLorc** *line*

Operand

line
specifies what you want saved in the LASTLORC buffer. If you omit line, the LASTLORC buffer is set to nulls.

Initial Setting

The initial setting of LASTLORC is a null string.

Notes for Macro Writers

LASTLORC does not have to be set explicitly. It is normally automatically updated with the last user subcommand when a LOCATE, CLOCATE, CHANGE, or any subcommand in the FIND family is executed.

LASTLORC provides a memory capability which can be used by a macro explicitly. For example, SCHANGE uses LASTLORC so that a user can repeat the last CHANGE or CLOCATE without having to reenter the data. Likewise a macro that issues a FIND or LOCATE might use this so that the user need not retype a command when they just want to continue a search.

Return Codes

0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET LINEND

Use the LINEND option to determine whether or not the editor is to recognize a pound sign (#) or other character as a line-end character.

Format

[SET]	**LINENd**	**ON** **OFF**	[*char*]

Operands

ON
: allows you to enter several adjacent subcommands, separated by the default character (a pound sign), or a specified character.

char
: is the special character to be used as a line-end character. Use special characters such as "@", "$", or "%". Other characters such as "/", a delimiter, or alphabetic characters may cause problems. The default line-end character is a pound sign (#).

OFF
: specifies that the editor is not to recognize a line-end character.

Initial Setting

```
LINEND  ON  #
```

Usage Notes

1. The line-end character may be specified in hexadecimal if SET HEX ON is in effect.
2. For consecutive linend characters (for example, FILE####) and for a single linend character which follows the last subcommand entered (for example, FILE#), XEDIT stacks a null line for each linend character minus 1.
3. In line mode the line-end character from CP will also be recognized during your XEDIT session. To set the CP linend character off or redefine it, issue:

   ```
   cp terminal linend off
           or
   cp terminal linend char
   ```

   with char as your choice of CP line-end character. (See the TERMINAL command in the *VM/SP CP General User Command Reference* for more information on the CP line-end symbol).
4. If SET CASE Uppercase is in effect, the command line is uppercased before it is scanned for LINEND characters. This can cause commands with case-sensitive targets to fail to work properly when issued in conjunction with a SET CASE M R command. For example, issuing a "SET CASE M R#FIND xxx" while SET CASE Uppercase is in effect causes the "xxx" to be uppercased before the editor searches for it.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

```
next 1#change /A/B/#TOP
```

executes as:

```
next 1
change /A/B/
top
```

SET LRECL

Use the LRECL option to define a new logical record length, which is used when the file is written to disk or directory.

Format

[SET]	**LRecl** *n* *

Operand

n

is the logical record length. For a file with a fixed record format (F), n is the length of each record. For a file with a variable record format (V), n is the maximum record length. If you specify an asterisk (*), the logical record length is set to the WIDTH as defined in the XEDIT (or LOAD) command.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Notes

1. The value (n) specified must be less than or equal to the value specified in the WIDTH operand of the XEDIT or LOAD subcommand (in the profile). (WIDTH is the amount of virtual storage used for any file line, regardless of its format on disk or directory.)
2. If the new logical record length is smaller than the previous one, data may be lost when the file is written to disk or directory. A smaller logical record length may also create new values for zone settings, the column pointer, and the truncation column. The following relationships should be verified:

   ```
   ZONE1 ≤ COLUMN POINTER ≤ ZONE2 ≤ TRUNC ≤ LRECL ≤ WIDTH
   ```

   The column pointer can also be positioned at TOL (Zone1 − 1) and at EOL (Zone2 + 1).
3. Refer to the LOAD subcommand for more details on the initial setting of LRECL during profile execution.

Messages

516E	LRECL too large for V-format file [RC=4]
519E	LRECL must be lower than WIDTH (*nn*) [RC=5]
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
560E	Not enough space for serialization between TRUNC and LRECL
698W	New record length may result in loss of double-byte characters [RC=3]

Return Codes

0	Normal
3	Records truncated
4	Lrecl must be lower than 65536 for recfm V
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET MACRO

Use the MACRO option to control the order in which the editor searches for subcommands and macros.

Format

[SET]	**MACRO**	**ON** **OFF**

Operands

ON
specifies that the editor is to look for macros before it looks for subcommands.

OFF
specifies that the editor is to look for subcommands before it looks for macros.

Initial Setting

```
MACRO  OFF
```

Usage Notes

1. This SET option can resolve an ambiguous situation, when a macro and a subcommand have the same name. If the subcommand has a synonym defined, the macro name executes only if SYNONYM is set OFF. (See also SET SYNONYM.)
2. The subcommand name SET is required with this option (to avoid conflict with the MACRO subcommand).

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET MASK

Whenever a new line is inserted in a file by a subcommand or macro, it is prefilled with the contents of a mask. Initially, the mask contains all blanks. You can use the MASK option to change the contents of the mask.

Format

[SET]	**MASK** **Define** **Immed** [*text*] **Modify**

Operands

Define
displays the scale in the command line to help you define a new mask. You can type the new mask over the scale. The size of the mask is limited by the width of the screen. (If you do not type a new mask over the scale, the scale becomes the new mask.)

Immed [*text*]
replaces immediately the current mask by the specified text. If no text is specified, it replaces the current mask with a blank line.

Modify
displays the current mask in the command line. You can *type over the mask* to redefine it. (The scale does not appear to assist you in defining the mask, as it does with the DEFINE operand.)

Initial Setting

The initial setting of the mask is a blank line.

Usage Notes

1. All subcommands and macros that insert lines in a file, prefill the new line with the mask. These subcommands are ADD, INPUT, and REPLACE, as well as the prefix subcommands A and I. The macros are SI and the prefix macro SI.
2. You can use the QUERY MASK subcommand to display the current mask in the message line, but you cannot change it.
3. When using SET MASK DEFINE or SET MASK MODIFY to reset the mask to a blank line, you can press the spacebar once and then press the ERASE EOF key; this prevents you from clearing the mask if you accidentally press ERASE EOF. Pressing only the ERASE EOF key does not clear the mask.

Note for Macro Writers

EXTRACT/MASK/ subcommand returns the current mask.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

A mask is analogous to a preprinted form that has the same information on every line, and you type in the variable information.

Figure 13 is an example of using SET MASK to define the "comments" area in a PLI program.

1. Enter the following subcommand:

   ```
   ====> set mask define
   ```

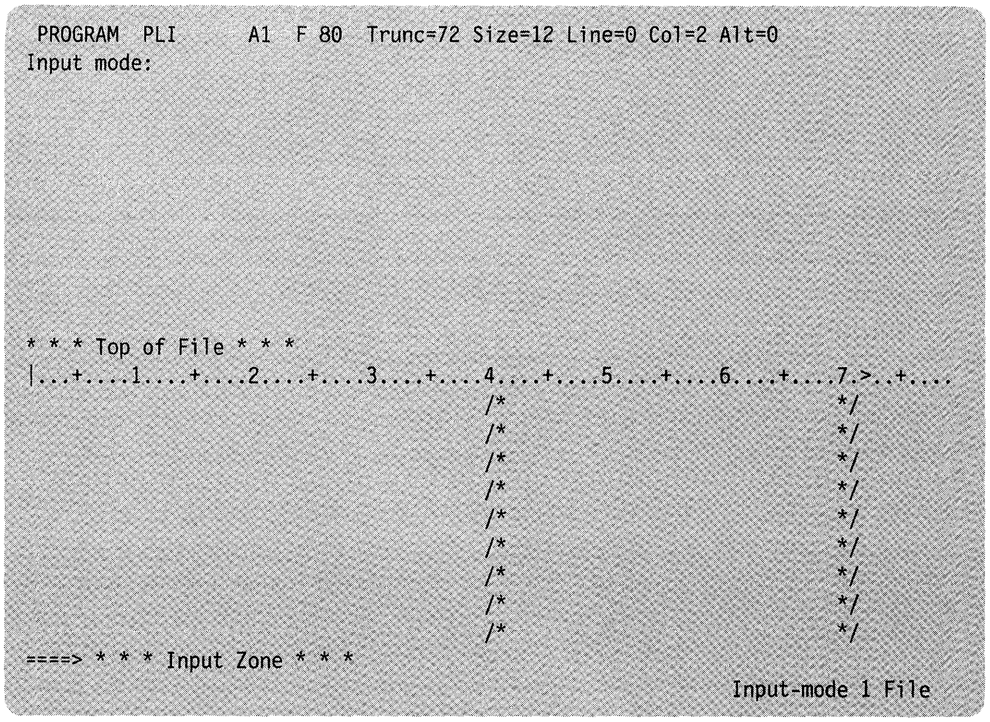
2. The scale appears in the command line:

   ```
   ====> ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.
   +....8....+....9....+...10....+...11....+...12....+...13... X E D I T  1 File
   ```

3. Type over the scale to define the mask, deleting any scale characters.

   ```
   ====>                                   /*                              */
                                                               X E D I T  1 File
   ```

4. If the INPUT subcommand is entered, each line in the input zone contains the mask:



```
 PROGRAM  PLI      A1  F 80  Trunc=72 Size=12 Line=0 Col=2 Alt=0
Input mode:









* * * Top of File * * *
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7.>..+....
                                        /*                           */
                                        /*                           */
                                        /*                           */
                                        /*                           */
                                        /*                           */
                                        /*                           */
                                        /*                           */
                                        /*                           */
                                        /*                           */
====> * * * Input Zone * * *
                                                            Input-mode 1 File
```

Figure 13. Using the SET MASK Subcommand

SET MSGLINE

Use the MSGLINE option to define the location of the message line on the screen, and the maximum number of lines that a message may occupy. It may also be used to determine whether or not a blank line is normally displayed for the message line.

Format

[SET]	**MSGLine** **ON** **OFF** M[+*n* |-*n*] | [+|-]*n* [*p* |1] [**Overlay**]

Operands

ON
allows you to receive messages at the screen location defined.

M[+*n*|-*n*]
stands for the middle of the screen (rounded up for odd-sized screens). M can be combined with a constant positive or negative integer to mean n lines below the middle of the screen (M + n) or n lines above the middle of the screen (M − n).

[+|-]*n*
indicates that the MSGLINE is located n lines from the top of the screen (+ is implied/specified) or from the bottom of the screen (− n).

p
is the maximum number of lines which can be used to display messages. The number of lines used to display messages will not exceed the number of lines in the actual messages. If the messages do not fit on p lines, the messages will be passed to CMS. See Usage Note 1 for information on displaying messages passed to CMS.

Overlay
indicates that you do not want a blank line on the screen for messages. If OVERLAY is specified, no message line is displayed unless a message is issued. When OVERLAY is specified, reserved lines (including the scale, tabline, etc.) and file lines are displayed wherever the message line was defined until a message is issued. If OVERLAY is not specified, a blank message line is displayed wherever the message line was defined. If the message line is defined as the same line as the command line, the command line overlays the blank MSGLINE so that a command line is available.

OFF
turns off the message line. All XEDIT messages are passed to CMS. See Usage Note 1 for information on displaying messages passed to CMS.

Initial Setting

```
MSGLINE ON 2 2
```

Usage Notes

1. If multiple messages that do not fit on the message line(s) need to be displayed, or MSGLINE is set OFF, messages are passed to CMS to be displayed.

   When full-screen CMS is ON, the CMSOUT window which is connected to the CMS virtual screen will contain the output. (Output will appear in the CMSOUT window by default; you can route the output to another window by using the CMS ROUTE command, explained in the *CMS Command Reference*). To see all of the information in the virtual screen, you can use the CMS SCROLL command. The screen will be cleared automatically when you scroll to the bottom of the file. An alternative way to clear the screen is to issue the CMS DROP WINDOW command. If you delete the CMSOUT window, you won't see messages that are passed to CMS.

   When full-screen CMS is OFF, the screen is cleared to display the message(s). Press the CLEAR key to redisplay the file. No alarm sounds on either the CMS screen or the next XEDIT screen.

2. The command line and MSGLINE may be the same line. If the messages overlay the command line you can press the CLEAR key (or any key that does not produce a message) to restore the command line.

3. When full-screen CMS is ON and messages are passed to CMS, the messages will be displayed with the XEDIT msgline color attributes—color, hilite, pss, etc.

Messages

520E	Invalid operand: *operand* [RC = 5]
521E	Invalid line number [RC = 5]
526E	Option *option* valid in display mode only [RC = 3]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
3	Subcommand valid only for display terminal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET MSGMODE

Use the MSGMODE option to control the message display.

The format of the SET MSGMODE subcommand is:

[SET]	**MSGMode** **ON** [**Short|Long**] **OFF**

Operands

ON
displays all messages at the terminal.

Long
displays all messages in their full length.

Short
Information (I) and warning (W) messages are not displayed. Error (E) messages are displayed as a NOT (¬) symbol. All other messages are displayed in their full length.

OFF
No messages or data is displayed.

Initial Setting

`MSGMODE ON LONG` (unless the NOMSG option was specified on the XEDIT command).

Usage Notes

1. When MSGMODE is set to OFF, messages are not displayed but are still generated internally (in long or short form, according to the setting). However, you can display the last message generated by using the following sequence:

   ```
   set msgmode on
   query lastmsg
   ```

   In a macro, EXTRACT/LASTMSG/ can be used to get the last message generated.

2. To enter XEDIT with the initial setting of MSGMODE OFF use the NOMSG option on the XEDIT or LOAD subcommand. Please refer to the XEDIT command in this publication for an explanation of the NOMSG option.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET NONDISP

Use the NONDISP option to define for XEDIT and CMS a character to be used in place of nondisplayable characters.

Format

[SET]	**NONDisp** [*char*]

Operand

char
specifies a character to be used. If not specified, a blank will be used.

Initial Setting

```
Defined by the CMS setting.
```

Usage Notes

1. The nondisplayable character may be specified in hexadecimal if SET HEX ON is in effect.
2. Setting a nondisplayable character to a visible one, for example, SET NONDISP ", is helpful when doing full screen changes on display terminals. When changing lines on the display, XEDIT tries to preserve nondisplayable characters within modified lines by comparing the line returned from the terminal with its old contents. This may lead to unexpected changes, particularly if the DELETE or INSERT keys are used.
3. The NONDISP character specified must be a displayable character. Nondisplayable characters are ignored.
4. If you issue CMS SET NONDISP while in XEDIT, the new nondisplayable character will not appear in a line containing nondisplayable characters until that line is altered. Similarly, if you issue CMS SET OUTPUT while in XEDIT, the changed character won't appear until the line containing the character(s) to be changed is altered.
5. The translation of the nondisplayable character depends upon the type of terminal and whether or not SET APL ON or SET TEXT ON is in effect. See the *VM/SP Terminal Reference* for the nondisplayable character translation tables.
6. Changing the editor NONDISP setting also changes the CMS NONDISP setting.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET NULLS

Use the NULLS option to specify whether trailing blanks in each line are written to the screen as blanks (X'40') or nulls (X'00').

Format

[SET]	**NULls** **ON** **OFF**

Operands

ON
specifies that all trailing blanks are to be replaced with nulls. This allows you to use the insert key on the IBM 3270 keyboard to insert characters in a line.

OFF
specifies that trailing blanks are not to be replaced with nulls. The insert key cannot be used to insert characters in a line unless the ERASE EOF key first is used to remove at least as many trailing blanks as characters to be inserted.

Initial Setting

```
NULLS  OFF
```

Usage Notes

1. You can use any key defined as "NULLKEY" instead of issuing SET NULLS ON if you wish to use the insert key for only one line. The "NULLKEY" function sets nulls on in the field that contains the cursor. If the cursor is on a prefix area, then blanks will be changed to nulls on the file line with which that prefix area is associated. If you move the cursor to another field and wish to use insert mode, you must press the key defined as "NULLKEY" again. "NULLKEY" is initially assigned by the editor to the PA2 key.
2. If either the WIDTH option on the XEDIT command or a SET VERIFY subcommand are set to a value less than the screen width, you can use the insert key to insert characters with SET NULLS OFF.
3. See also SET FULLREAD.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET NUMBER

Use the NUMBER option to specify whether or not file line numbers are to be displayed in the prefix area.

Format

[SET]	**NUMber** **ON** **OFF**

Operands

ON
causes a five-digit line number to be displayed in the prefix area of the lines. The line numbers are sequential and are recomputed when lines are added or deleted.

OFF
causes five equal signs (= = = = =) to be displayed in the prefix area of each line, unless the prefix area has been set to NULLS. (Refer to SET PREFIX in this publication.)

Initial Setting

```
NUMBER  OFF
```

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET PAn

Use the PAn option to define a meaning for a specified hardware attention (PA) key or to remove the meaning associated with the specified PA key.

Format

```
[SET]     PAn   [BEFORE]   [string ]
                [AFTER ]   [NULLKEY]
                [ONLY  ]   [COPYKEY]
                [IGNORE]   [TABKEY ]
```

Operands

n
: specifies a PA key number (1,2, or 3). If SET PAn is issued (without an additional operand), the meaning associated with that PA key is removed.

BEFORE
: specifies that the key definition is executed before the contents of the command line. BEFORE is the default for all the keys, unless the string begins with a "?" or "=." For these two strings, ONLY is the default.

AFTER
: specifies that the key definition is executed after the contents of the command line.

ONLY
: specifies that only the key definition is executed, thereby ignoring the command line.

IGNORE
: specifies that the key definition is ignored when something is entered on the command line, thereby only executing the command line.

string
: is any XEDIT subcommand or macro (including CP, CMS, or EXEC), which is executed when the PA key is pressed. If string is null, the PA key will be undefined.

NULLKEY
: When the corresponding PA key is pressed, blank characters are changed to nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, then blanks will be changed to nulls on the file line with which that prefix area is associated.

TABKEY
: When the corresponding PA key is pressed, the cursor is moved into the next tab column, as defined by the SET TABS subcommand.

COPYKEY
: When the corresponding PA key is pressed, the exact content of the virtual screen is copied into the printer spool.

Initial Setting

```
SET PA1   BEFORE COMMAND CMS POP WINDOW WM
SET PA2   BEFORE NULLKEY
SET PA3   ONLY ?
```

Usage Notes

1. You can assign a sequence of subcommands to a single PA key by: setting off the LINEND character; setting the PA key to the subcommands separated by LINEND characters; then setting the LINEND character back on before using the PA key.

   For example:

   ```
   set linend off
   set pa2 next#c/A/B
   set linend on #
   ```

2. If a PA key set to TABKEY is pressed and there are no tab columns available on the screen, the position of the cursor remains unchanged.
3. PA keys may be used in input mode. (See the "Usage Notes" section of the INPUT subcommand.)
4. When you use a PA key set to COPYKEY, you should close the printer at the end of the editing session. If your printer is defined as a 3203, 3211, 3262, 4245, or 3289e, refer to the *VM/SP CP General User Command Reference* under LOADVFCB for information on the default FCB.
5. When you press a PA key set to TABKEY, COPYKEY, or NULLKEY, no screen changes are processed including the command line and the keywords, BEFORE, AFTER, ONLY, and IGNORE, have no effect.
6. Setting FULLREAD ON will prevent you from losing any screen changes when you press a PA key and a message is displayed on a cleared screen.
7. The setting of the CP break key (PA1 by default) overrides any later defined editor setting unless SET BRKKEY OFF is specified.

Messages

520E	Invalid operand: *operand* [RC=5]
525E	Invalid {PFkey|PFkey/PAkey} number [RC=5]
545E	Missing operand(s) [RC=5]
614E	Screen modifications lost. See SET FULLREAD to use PAkeys safely. [RC=8]
622E	Insufficient [free] storage for copykey [RC=104]
657E	Undefined PFkey/PAkey

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Modifications lost because PA key pressed when message pending
104	No storage is available

SET PACK

Use the PACK option to specify whether or not the editor is to pack the file when it is written to disk or directory.

Format

[SET]	PACK ON OFF

Operands

ON
specifies that the editor is to compress records in a file so that they can be stored in packed format when a FILE or SAVE subcommand is issued.

OFF
specifies that the editor is not to pack the file when it is written to disk or directory.

Initial Setting

According to the format of the file edited. PACK ON if the file is in packed format; PACK OFF if the file is not in packed format.

Usage Notes

1. When an XEDIT command is issued for a file in packed format on disk or directory, the editor automatically issues a SET PACK ON subcommand. Thus, when the file is filed or saved, it will be in packed format when it is written back to disk or directory.
2. The PACK option is compatible with the PACK option of the CMS COPYFILE command. A file can be packed (or unpacked) using either the editor SET PACK subcommand or the CMS COPYFILE command.
3. A file in packed format should not be modified in any way while it is on disk or directory (for example, a PUT subcommand would append data to a file while it is on disk or directory). If such a file is modified, neither XEDIT nor the CMS COPYFILE command will be able to reconstruct the file.

Responses

The editor displays the record format in the IDLINE as:

FP (fixed packed)
or
VP (variable packed)

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET PENDING

Use the PENDING option to control the execution of a prefix macro and the status of the screen while the prefix macro is being executed. This option, designed to be issued from a macro, can be used to display a prefix subcommand or macro in the prefix area of the current line, or to indicate to the user that a prefix subcommand or macro that was entered is the beginning or end of a block. In both cases, the screen is placed in a pending status (described in the notes below). SET PENDING can also be used to notify the user that a prefix macro was entered incorrectly.

Format

[SET]	**PENDing**	**ON**	*string*
		BLOCK	*string*
		ERROR	*string*
		OFF	

Operands

ON *string*
displays "string", which can be any prefix subcommand or macro, in the prefix area of the current line and displays a pending notice in the status area (described in the notes below). By default, the string in the prefix area is highlighted. This form of SET PENDING adds an entry to the "pending list" (list of pending prefix subcommands and macros), which is described in the notes below.

BLOCK *string*
displays "string", which is the block form of any prefix subcommand or macro (for example, DD is the block form of the D prefix subcommand) in the prefix area of the line in which it was entered and displays a pending notice in the status area (described in the notes below). By default, the string in the prefix area is highlighted. This form of SET PENDING is used to notify the user that the beginning (or end) of a block was entered. The string would normally be the same as that entered by the user, as determined by the macro. This form of SET PENDING adds an entry to the "pending list," which is described in the notes below.

ERROR *string*
specifies that the string is to be displayed in the prefix area, prefixed by a question mark (?). By default, the string in the prefix area is highlighted. This indicates that an error occurred while a prefix macro (or subcommand) was executing, for example, if the macro was entered incorrectly. The string would normally be the incorrectly-entered prefix macro, as determined by the prefix macro, so that the user could correct the error.

Because this entry in the pending list is deleted the next time the list is processed, pressing the ENTER key while "? string" is displayed resets the prefix area. (This prevents subsequent attempts by the user to execute an incorrect prefix subcommand or macro.) This form of SET PENDING does not cause a pending notice to be displayed.

OFF

removes a pending prefix subcommand or macro from the prefix area of the current line (that is, removes this entry from the pending list), and resets the prefix area and the status area.

Notes for Macro Writers

1. The "pending list" is a list of prefix subcommands and macros that have not yet been executed. Every time the editor reads the screen, the pending list is updated with any new prefix subcommands and macros, each of which causes an entry to be added to the list. Each entry is associated with a specific line in the file. The pending list is executed when it is updated. If a prefix macro returns a non-zero return code, execution of the pending list stops and all entries not executed remain pending, until the user presses a key.

   An entry is deleted from the pending list when it is executed or overtyped on the user's screen with a new prefix subcommand, prefix macro, or blanks. An entry can also be added by using "SET PENDING ON string" and deleted by issuing "SET PENDING OFF" or the RESET subcommand.

   A prefix macro can control its execution and the screen status by examining information in the pending list (by using EXTRACT /PENDING. . ./) and by examining the *argument string* that is automatically passed to a prefix macro when it is invoked. The argument string contains information pertinent to the prefix macro when it is called, for example, the file line of the prefix area in which it was entered. For a description of this argument string, see the *VM/SP System Product Editor User's Guide*. The macro can then take appropriate action, which may include using some form of SET PENDING.

2. On display terminals, the pending status is indicated by the following notice in the status area:

   ```
   'value' pending...
   ```

   where "value" is the name of the pending prefix macro or subcommand that was entered in the prefix area. If multiple prefix subcommands or macros are pending, the first one (starting from the top of file) is displayed in the pending notice.

3. After all the prefix subcommands and macros in the pending list are processed, the current line is restored to what it was before processing. Therefore, a prefix macro does not usually need to be concerned with restoring the current line. However, a macro can issue "SET PENDING ON /" to specify which line is current when it is finished executing. (When multiple "/" prefix subcommands are entered, the last one executed sets the current line.)

4. When a macro or subcommand is first put in the pending list, it is checked (via the CMS STATE command) to see if it exists. If it does not exist, it is left pending. Creating the macro will not cause the pending macro to be automatically executed. You must reenter it into the prefix area for it to be executed.

5. When the pending list is executed it is scanned for SCALE, TABL, and / prefix subcommands after all the other prefix subcommand and macros have been executed. SCALE, TABL, and / are executed in the logical screen in which they were issued (not just on the screen in which the pending list was executed). Because they are executed after the rest of the pending list is processed, a prefix macro may specify them and they may be executed before the screen is redisplayed (this is particularly useful for setting the current line from a prefix macro). When multiple SCALE, TABL, and / prefix subcommands are issued,

only the last one issued is used. If the last TABL or SCALE is specified on a line that does not fall on the screen, then it is ignored.

6. For more information on the pending list and writing prefix macros, see the subcommands EXTRACT/QUERY PENDING, and SET/EXTRACT/QUERY PREFIX SYNONYM. For information on IBM-supplied prefix macros, see "Chapter 4: Prefix Subcommands and Macros." For tutorial information on writing prefix macros and examples of how to use SET PENDING in prefix macros, see the *VM/SP System Product Editor User's Guide.*

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

```
set pending on A
```

adds an entry to the pending list and displays A==== in the prefix area of the current line and the notice "'A' pending. . ." in the status area.

```
:3 set pending off
```

removes the current line from the pending list, that is, makes line 3 the current line and resets the prefix area and the status area.

```
:10 set pending error XX20
```

makes line 10 the current line and displays ?XX20 in the prefix area. (XX20 is an invalid form of the X prefix macro.)

```
:100 set pending block XX
```

makes line 100 the current line and displays "XX" in the prefix area (highlighted), and displays "'XX' pending. . ." in the status area.

Assuming this was issued from a prefix macro note that :100 is an absolute line number target. It is used to make this line current for the SET PENDING subcommand, which operates on the current line. Because the current line is restored after the pending list is executed (see notes, above), the block form of the macro is displayed in the prefix area of the line in which it was entered (which is not necessarily the current line).

Therefore, if "XX" is entered on line 100 of the file, and line 100 is not the current line, :100 makes line 100 current for the SET PENDING subcommand. However, because the current line is restored after execution, the user sees "XX" displayed in line 100, where it was entered.

SET PFn

Use the PF*n* option to define a meaning for a specified hardware program function (PF) key or to remove the meaning, if any, associated with the specified PF key.

Format

```
[SET]    PFn   ┌BEFORE┐   ┌string ┐
               │AFTER │   │NULLKEY│
               │ONLY  │   │COPYKEY│
               └IGNORE┘   └TABKEY ┘
```

Operands

n
specifies a PF key number (n can be 1 through 24). If SET PFn is issued (without an additional operand), the meaning associated with that PF key is removed.

BEFORE
specifies that the key definition is executed before the contents of the command line. Before is the default for all the keys, unless the string begins with a "?" or "=." For these two strings, "ONLY" is the default.

AFTER
specifies that the key definition is executed after the contents of the command line.

ONLY
specifies that only the key definition is executed, thereby ignoring the command line.

IGNORE
specifies that the key definition is ignored when something is entered on the command line, thereby only executing the command line.

string
is any XEDIT subcommand or macro (including CP, CMS, or EXEC), which is executed when the PF key is pressed. If string is null, the PF key will be undefined.

NULLKEY
When the corresponding PF key is pressed, blank characters are changed to nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, then blanks will be changed to nulls on the file line with which that prefix area is associated.

TABKEY
When the corresponding PF key is pressed, the cursor is moved into the next tab column, as defined by the SET TABS subcommand.

COPYKEY
When the corresponding PF key is pressed, the exact content of the virtual screen is copied into the printer spool.

Initial Setting

```
SET PF01 BEFORE HELP MENU
SET PF02 BEFORE SOS LINEADD
SET PF03 BEFORE QUIT
SET PF04 BEFORE TABKEY
SET PF05 BEFORE SCHANGE 6
SET PF06 ONLY   ?
SET PF07 BEFORE BACKWARD
SET PF08 BEFORE FORWARD
SET PF09 ONLY   =
SET PF10 BEFORE RGTLEFT
SET PF11 BEFORE SPLTJOIN
SET PF12 BEFORE CURSOR HOME
```

Note: These are the initial settings. On a terminal equipped with 24 PF keys, PF keys 13 through 24 have the same values as PF keys 1 through 12 as discussed here, except PF key 17 which is set to BEFORE SCHANGE 18.

Usage Notes

1. You can assign a sequence of subcommands to a single PF key by: setting off the LINEND character; setting the PF key to the subcommands separated by LINEND characters; then setting the LINEND character back on before using the PF key.

   For example:

   ```
   set linend off
   set pf3 next#c/A/B
   set linend on #
   ```

2. If a PF key set to TABKEY is pressed and there are no tab columns available on the screen, the position of the cursor remains unchanged.

3. To get the same effect as a CP SET PF DELAY, use:

   ```
   set pfn cmsg...
   ```

4. PF keys may be used in input mode. (See the "Usage Notes" section of the INPUT subcommand.)

5. When you use a PF key set to COPYKEY, you should close the printer at the end of the editing session. If your printer is defined as a 3203, 3211, 3262, 4245, or 3289e, refer to the *VM/SP CP General User Command Reference* under LOADVFCB for information on the default FCB.

6. When you press a PF key set to TABKEY, COPYKEY or NULLKEY, no screen changes are processed including the command line, and the keywords BEFORE, AFTER, ONLY and IGNORE have no effect.

Messages

520E	Invalid operand: *operand* [RC=5]
525E	Invalid {PFkey|PFkey/PAkey} number [RC=5]
545E	Missing operand(s) [RC=5]
622E	Insufficient [free] storage for copykey [RC=104]
657E	Undefined PFkey/PAkey

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
104	No storage is available

SET POINT

Use the POINT option to define a symbolic name for the current line. More than one name can be defined for a line by issuing separate SET POINT subcommands. You can use these names to refer to the line in subsequent target operands of XEDIT subcommands.

Format

[SET]	**Point** *.symbol* [**OFF**]

Operands

.symbol
is a symbolic name for the current line. The name must begin with a period and be followed by from one to eight alphanumeric or special characters, for example, .AAA.

OFF
deletes the specified symbol, without moving the line pointer. The symbol to be deleted must be specified in front of this operand.

Usage Notes

1. The POINT option makes it unnecessary for you to remember or to look up the line number of a line. A line can be referenced by its name at any time during the editing session. For example, if the following subcommand is issued:

   ```
   set point .XAVIER
   ```

   the current line is assigned the specified name.

   The name can then be referenced at any time during the editing session. For example:

   ```
   move 3 .XAVIER
   ```

   moves three lines, beginning with the current line, after the line named .XAVIER.
2. The line number of a line can change during an editing session; for example, adding lines before a particular line increments its line number. The symbolic name, however, stays with a line for the entire editing session.
3. The .xxxx prefix subcommand can also be used to assign a symbolic name to a line. In this case, the symbolic name is limited to four characters.
4. You can use the QUERY POINT subcommand to display the symbolic name(s) of the current line. You can use QUERY POINT * to display all symbolic names and their line numbers as they have been defined.

Notes for Macro Writers

The EXTRACT/POINT/ subcommand returns the symbolic name(s) and line number of the current line and EXTRACT/POINT */ returns the symbolic names and their line numbers in the file.

Messages

520E	Invalid operand: *operand* [RC=5]
539E	Named line not found [RC=2]
540E	Name already defined on line *nn* [RC=1]
541E	Invalid name [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
1	Duplicate name defined
2	Name does not exist for OFF function
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET PREFIX

Use the PREFIX option to control the display of the prefix area, or to define a synonym for a prefix subcommand.

Format

[SET]	**PREfix**	**ON**	**[Left|Right]**	
		OFF		
		Nulls	**[Left|Right]**	
	PREfix	**Synonym**	*newname*	*oldname*

Operands

ON
displays a five-character prefix area (= = = = =) for each logical line on the screen. The prefix area can be displayed on either the left (LEFT operand) or right (RIGHT operand) side of the screen. Prefix subcommands and macros can then be entered in the prefix area.

OFF
removes the prefix area from the screen.

Nulls
displays nulls in the prefix area enabling the use of the insert key.

Synonym
is the keyword operand that indicates a synonym is to be defined for a prefix subcommand or macro.

newname
is a synonym to be assigned to a prefix subcommand or macro. The new name can be up to five alphabetic or special characters.

oldname
is the name of a prefix subcommand or macro for which you are defining a synonym. The oldname can be up to eight characters.

Initial Setting

```
PREFIX ON LEFT
```

There are eight prefix synonyms defined.

>	PRFSHIFT
>>	PRFSHIFT
<	PRFSHIFT
<<	PRFSHIFT
X	PREFIXX
XX	PREFIXX
S	PRFSHOW
.....	SI

Usage Note

When using SET PREFIX ON with SET NUMBER ON, a five-digit line number is displayed instead of equal signs. When using SET PREFIX NULLS with SET NUMBER ON, any leading zeros are translated to nulls.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
548E	Invalid synonym operand: *operand* [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET RANGE

Use the RANGE option to define new limits for line pointer movement. In effect, this option defines a new top and bottom for the file. During the editing session, all subcommands (except for FILE and SAVE) operate only within the range.

Format

[SET]	**RANge** *target* 1 *target* 2

Operands

target1
is the top of the range.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

target2
is the bottom of the range.

Initial Setting

RANGE 1 n where n is equal to the size of the file.

Usage Notes

1. After the range is defined, the TOP subcommand moves the line pointer to the line that precedes target1 (to allow insertion at the top of the range). Target1 becomes the equivalent of the first line in the file. Top of Range becomes the equivalent of the Top of File line. The BOTTOM subcommand moves the line pointer to target2. Target2 becomes the equivalent of the last line in the file. End of Range becomes the equivalent of the End of File line.
2. If the SET RANGE option is entered while the line pointer is outside the limits being defined, the current line becomes the first line of the new range.
3. FILE and SAVE subcommands write the *entire* file to disk or directory. No other subcommands have access to data outside the range.
4. The following subcommand resets the range to the physical top and bottom of the file:

   ```
   set range -* *
   ```

   In order to set a range outside the limits of the current range, you must specify target1 and target2 as absolute line numbers. An alternate method is to issue:

   ```
   set range :1 *
   ```

   (See Appendix B, "Effects of Selective Line Editing Subcommands" on page 405 for information about SET RANGE and the SCOPE setting.)
5. SET RANGE cannot be issued when prefix subcommands or macros are pending and it cannot be issued from a prefix macro.

Messages

520E	Invalid operand: *operand* [RC=5]
528E	Invalid range: target2 (line *nn*) precedes target1 (line *nn*) [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
560E	Not enough space for serialization between TRUNC and LRECL
588E	Prefix subcommand waiting... [RC=8]

Return Codes

0	Normal
2	Target not found
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Prefix subcommand or macro waiting

SET RECFM

Use the RECFM option to define the record format for the file.

Format

[SET]	**RECFm** **F** **V** **FP** **VP**

Operands

F
specifies that the record format is fixed.

V
specifies that the record format is variable.

FP
specifies that the record format is fixed packed.

VP
specifies that the record format is variable packed.

Initial Setting

Based on the file type. See Appendix A, "File Type Defaults" on page 403.

Usage Note

When the record format is FP or VP, a SET PACK ON is automatically executed.

Messages

515E	RECFM must be F, V, FP, or VP [RC=5]
516E	LRECL too large for V-format file [RC=4]
520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
4	Lrecl must be lower than 65536 for recfm V
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET REMOTE

Use the REMOTE option to control the way XEDIT and CMS handle the display in terms of data transmission.

Format

[SET]	**REMOte**	**ON** **OFF**

Operands

ON
specifies that XEDIT is to compress the screen by removing nulls and combining data when five or more of the same characters occur consecutively in a data stream. This minimizes the amount of data transmitted and shortens the buffer, thus speeding transmission.

OFF
specifies that XEDIT is not to compress the screen. Data will be transmitted with no minimization.

Initial Setting

```
Defined by the CMS setting.
```

Usage Note

Changing the editor REMOTE setting also changes the CMS REMOTE setting.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET RESERVED

Use the RESERVED option to reserve a specified line on the screen, thereby preventing the editor from using that line. The line can be used for displaying blank or specified information with or without the following features:

- Color
- Extended highlighting
- Programmed symbol set
- Highlighting.

The RESERVED option can also be used to give a reserved line back to the editor. This option is designed to be issued from a macro.

Format

```
[SET]   RESERved   M [+n |-n]   [color] [exthi] [PSs]  High   [text]
                   [± | -]n                            Nohigh
                                Off
```

Operands

M[+*n*|-*n*]
specifies the line which is to be the reserved line. M stands for "middle of the screen" (rounded up for odd sized screens). M may be combined with a constant positive or negative integer to mean n lines below the middle of the screen (M+n), or n lines above the middle of the screen (M−n). For example, RESERVED M means the "middle of the screen," and RESERVED M+3 means "three lines below the middle of the screen."

[±|-]*n*
specifies the line which is to be the reserved line. n or +n (+ is implicit) specifies that the reserved line is displayed n lines from the top of the screen; −n specifies that the reserved line is displayed n lines from the bottom of the screen. For example, RESERVED −3 means that the reserved line is three lines from the bottom of the screen.

color
color can be any one of the following:

```
Blue      Turquoise
Red       Yellow
Pink      White
Green     Default.
```

If you omit color, the default is DEFAULT (default base color).

exthi
extended highlighting can be any one of the following:

```
BLInk
REVvideo  (reverse video)
Underline
NONe.
```

If you omit *exthi*, the default is NONE (no extended highlighting).

PSs
specifies the character set to be used. PS0 indicates that the default character set will be used. PSs can be any of the following:

```
PS0      PSD
PSA      PSE
PSB      PSF
PSC.
```

High
indicates that the data in the reserved line is to be highlighted.

Nohigh
indicates that the data in the reserved line is not to be highlighted (normal intensity).

text
specifies the information that is to be displayed in the reserved line.

Off
indicates that a line reserved previously is to be returned for use by the editor.

Initial Setting

No reserved lines are defined.

Usage Note

The QUERY RESERVED subcommand may be used to display the line numbers of reserved lines.

Notes for Macro Writers

1. The EXTRACT/RESERVED/ subcommand returns line numbers of reserved lines. The EXTRACT/RESERVED */ subcommand returns information about reserved lines.

2. Reserved lines are associated with the file being edited when they are defined. They are displayed only when that file is being displayed.

3. SET RESERVED +n and SET RESERVED −n are considered to be two separate lines, even when they point to the same line on the screen.

   For example, assume you have a 43-line screen and issue the subcommands SET RESERVED 21 and SET RESERVED −4. Two different lines are reserved on this screen: line 21 and line 40. However, if the screen size changes to a 24-line screen, both reserved lines will fall on the same line. In this case, the editor keeps both definitions but displays only the last one defined.

   When turning a reserved line off, you must specify the line the same way you defined it.

4. If you reduce the logical screen size (for example, by splitting the screen), only those reserved lines that fall within the smaller screen size are displayed. The definition of lines that do not fit on the screen are kept, even though they are not displayed.

5. A SET RESERVED subcommand issued for the first line of the command line will be ignored, unless SET CMDLINE OFF has been issued. This prevents you from accidentally losing the command line. A reserved line can be set on the second line of the command line if CMDLINE ON.

6. For information on defining various attributes for fields within a reserved line, see the SET CTLCHAR subcommand.

7. The SET RESERVED subcommand accepts the color, extended highlighting, and programmed symbol set operands whether or not the device has the ability to use those attributes. However, the action taken depends upon the device. For example, HIGH and NOHIGH are ignored on a 3279 display (unless the color was DEFAULT), color is ignored on a 3278 display, and color, extended highlighting, and character set are ignored on a 3277 display. Also, QUERY and EXTRACT return all the settings, even those that may be ignored on any given terminal.

8. On 3270 terminals equipped with the Programmed Symbol (PS) feature you can specify alternate character sets to be used on your terminal. The characters in these sets use different symbols, such as a different style or font, than the default character set.

   New character sets are loaded using the Graphic Data Display Manager (GDDM) program product (5748-XXH) or an application that loads programmed symbol sets. Character sets should be loaded before invoking XEDIT. If a new set is loaded, it will not be recognized and the last character set loaded prior to invoking the editor will be displayed. Likewise, if you drop a programmed symbol set, the editor is not aware that the set is no longer available and attempts to use that set. This may cause I/O errors when the display is written (resulting in SET TERMINAL TYPEWRITER). To avoid problems, do not load or delete program symbol sets when in the XEDIT environment.

9. A SET RESERVED subcommand issued for the first line of the message line will be ignored unless SET MSGLINE OVERLAY has been issued.

10. If you use SET RESERVED for screen row one, the reserved line will be offset by one column; the row will begin in column two instead of column one.

Response

The information specified in the text operand, if any, appears on the specified reserved line.

Messages

520E	Invalid operand: *operand* [RC=5]
521E	Invalid line number [RC=5]
526E	Option *option* valid in display mode only [RC=3]
533E	Line *nn* is not reserved [RC=4]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
3	Operand is valid only for display terminal
4	Line is not reserved
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SCALE

Use the SCALE option to display a scale line under the current line (the default) or on a specified line, to assist you in editing.

Format

[SET]	**SCALe**	**ON** **OFF**	[M[+*n* |-*n*] | [± | -]*n*]

Operands

ON
displays the scale on the screen.

M[+*n*|-*n*]
specifies the line in which the scale line is displayed. M stands for "middle of the screen" (rounded up for odd sized screens). M may be combined with a constant positive or negative integer to mean n lines below the middle of the screen (M+n), or n lines above the middle of the screen (M−n). For example, SCALE ON M means the "middle of the screen," and SCALE ON M+3 means "three lines below the middle of the screen."

[±|-]*n*
specifies the line in which the scale line is displayed. n or +n (+ is implicit) specifies that the scale line is displayed n lines from the top of the screen; −n specifies that the scale line is displayed n lines from the bottom of the screen. For example, SCALE ON −3 displays the scale line three lines from the bottom of the screen.

OFF
removes the scale from the screen.

Initial Setting

```
SCALE ON M+1.
```

Usage Notes

1. The scale looks like this:

```
<...+..|.1....+....2....+....3.>..+....4T...+....5....+....6....+....7...
.      .                         .      .
.      .                         .      .
.      .column pointer           .      .
.                                .      .truncation column
.left zone                       .right zone
```

2. If a line occupies more than one screen line and the scale is on an adjacent line, the scale is not displayed for that line. The scale is redisplayed when the line pointer moves to a file line that occupies only one screen line.

Messages

520E	Invalid operand: *operand* [RC=5]
521E	Invalid line number [RC=5]
526E	Option *option* valid in display mode only [RC=3]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
3	Operand is valid only for display terminal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SCOPE

Use the SCOPE option to specify the set of lines on which the editor operates. The operand (ALL or DISPLAY) indicates if the collection includes all lines in the file or only the lines displayed. For an illustration of how SET SCOPE can be used effectively in combination with the other selective line editing subcommands (SET DISPLAY, SET SHADOW, and SET SELECT), see SET SELECT in this publication.

Format

[SET]	**SCOPE** **Display** **All**

Operands

Display
indicates that the editor acts only on those lines currently in the display range as specified by SET DISPLAY. It performs as if the lines which are outside the display range are not part of the file.

All
indicates that the editor acts on the entire file.

Initial Setting

```
SCOPE DISPLAY
```

Usage Notes

1. Target searches are performed only on lines within the current scope. If SCOPE DISPLAY has been set, the target search only considers and looks at displayed lines. This is true for all types of targets: absolute line numbers, a relative displacement from the current line, a line name, a simple string expression, or a complex string expression. Line movement by use of targets is done as if lines outside the scope had been removed. For example, NEXT or +1 may go from line 20 to line 40 if lines 21 to 39 are outside the display range.

2. If SCOPE DISPLAY is set and the current line's selection level is not within the SET DISPLAY n1 n2 values, the editor forces the current line out of the display, causing the first line following it in the scope to be the new current line. This can occur if:

   a. You issue SET SELECT changing the current line's selection level

   b. You issue SET DISPLAY, changing lines that are displayed

   c. You issue SET SCOPE DISPLAY and the previous setting was SCOPE ALL.

3. If SCOPE ALL is set, the current line is always included in the display. Its selection level is not modified, but its display status is forced as long as it remains the current line.

4. While SCOPE ALL is set, the editor acts on the entire file, even that portion of the file that is not displayed. Therefore, a line which is not displayed may be changed by the editor.

5. For examples of this and other selective line editing subcommands, you can examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display. See also Appendix B, "Effects of Selective Line Editing Subcommands" on page 405.
6. SORT, SET RANGE, and ALL work outside the scope.
7. If SCOPE DISPLAY is set and the last line of the file is a shadow line, with the current line being the end of file line, entering input mode will cause the last line of the shadow line group to be displayed as the current line. In this case, a line that was outside the scope can be changed.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SCREEN

Use the SCREEN option to divide the virtual screen into a specified number of logical screens so that you may edit multiple files or multiple views of the same file. Each logical screen becomes, in effect, an independent terminal with its own file identification line, command line, and message line.

Format

[SET]	**SCReen**	*n* [**<u>Horizontal</u>** | **Vertical**]
		Size *s* **1** [*s* **2** [*s* **3** ... [*sn*]]]
		Width *w***1** [*w***2** [*w* **3** ... [*wn*]]]
		Define *sl***1** *sw***1** *sh***1** *sv***1** [*sl***2** *sw***2** *sh***2** *sv***2**]...

Operands

n
: specifies the number of logical screens that the virtual screen is to be divided into. If only n is specified, Horizontal is assumed.

<u>Horizontal</u>
: specifies that the logical screens are arranged horizontally, that is, one on top of the other. n must be specified such that all horizontal screens are at least five lines long.

Vertical
: specifies that the logical screens are arranged vertically, that is, next to each other, from left to right. n must be specified such that all vertical screens are at least 20 columns wide.

Size *s1[s2[s3...[sn]]]*
: specifies that the screens are created horizontally, where "sn" is the number of lines in each logical screen. Any number of screens can be created, provided each is at least five lines long (that is, "sn" cannot be less than five).

Width *w1[w2[w3...[wn]]]*
: specifies that the screens are created vertically, where "wn" is the number of columns in each screen. Any number of screens can be created, provided each is at least 20 columns wide (that is, "wn" cannot be less than 20).

Define *sl1 sw1 sh1 sv1 [sl2 sw2 sh2 sv2]...*
: indicates that each screen is created according to the layout specified, where:

- "sln" is the number of lines in the logical screen.
- "swn" is the number of columns in the logical screen.
- "shn" is the line number of the upper left corner of the logical screen on the virtual screen.
- "svn" is the column number of the upper left corner of the logical screen on the virtual screen.

The maximum number of screens that can be created is 16, provided "sln" is not less than 5 and "swn" is not less than 20.

Initial Setting

`SCREEN SIZE` *s1*, where s1 is the number of rows in the virtual screen.

Usage Notes

1. When multiple files are being edited, the files are arranged in a "ring" in virtual storage (see the XEDIT subcommand for more information on the ring of files). If a SET SCREEN subcommand increases the number of logical screens, the additional screens are immediately filled with files selected from the ring of files.

   The file that immediately follows (in the ring) the last file that is displayed on the screen fills up the first empty logical screen, and so forth.

   Any files that were already displayed before the SET SCREEN subcommand was executed remain on the screen and keep their relative positions on the virtual screen.

   However, if the number of logical screens is decreased due to a SET SCREEN subcommand, files will still be displayed, beginning with the file in the ring that issued the SET SCREEN subcommand, as long as logical screens are available. Those files for which logical screens are no longer available are removed from the display.

2. When vertical screens are defined (using SET SCREEN WIDTH or SET SCREEN DEFINE), the entire width of the virtual screen must be accounted for, that is, the logical screens must occupy the full virtual screen width.

   When defining vertical screens, remember to consider the width of the logical screens compared with the line length of the file(s) being edited. If the file line length exceeds the logical screen width, lines are displayed up to that width, that is, they do not "wrap," and changing the SET VERIFY setting will not cause them to wrap.

   On vertical screens, messages are broken up into as many lines as needed to display the entire message (up to the number of lines specified in the SET MSGLINE subcommand). If a message exceeds this number, it is passed to CMS and the message is displayed. When full-screen CMS is OFF, press the CLEAR key to redisplay the file. When full-screen CMS is ON, the message will appear in the CMSOUT window. To see all of the information in the virtual screen, you can use the CMS SCROLL command. The screen will be cleared automatically when you scroll to the bottom of the virtual screen. An alternative way to clear the screen is to issue the CMS DROP WINDOW command. If you have deleted the CMSOUT window, you won't see messages that are passed to CMS.

3. The SET SCREEN subcommand retains the CURLINE, SCALE, TABLINE, CMDLINE, MSGLINE, and RESERVED locations on the screens, provided that these settings fall within the new screen size. Otherwise, the default settings are used.

   When CMDLINE ON is in effect and vertical screens are defined, the command line is displayed on the bottom line of the screen since lines cannot extend on multiple screen lines (see usage note 2). The status area is not displayed. The editor remembers this change and returns the CMDLINE to ON if the screen definition is subsequently changed to a non-vertical screen(s).

4. Entering input mode on one screen (issuing the INPUT subcommand without any operands), causes subcommands entered on other logical screens to either 1) be ignored, if the screen contains a view of the same file, or 2) remain on the command line and not execute until input mode is exited, if the screen contains a view of a different file.

5. For information on the order of processing, please refer to the publication, *VM/SP System Product Editor User's Guide*, Chapter 5.
6. If more than one logical screen is defined before issuing a "disconnect" from XEDIT, the screen setting will be redefined to one screen when the session is reconnected on a different sized terminal. This would be the equivalent of issuing a `SET SCREEN` 1 after reconnecting.
7. If the window being used is changed through the use of the XEDIT window option while there are multiple logical screens, the editing session will continue with only one logical screen (with the same number of rows and columns as the virtual screen).

Responses

In horizontal screens, the status area of each logical screen contains the number of files being edited.

The screens are displayed as specified.

Messages

520E	Invalid operand: *operand* [RC = 5]
526E	Option *option* valid in display mode only [RC = 3]
534E	Too many logical screens defined [RC = 4]
536E	Logical screens exceed virtual screen size [RC = 1]
537E	Each logical screen must contain at least 5 lines and 20 columns [RC = 4]
543E	Invalid number: *number* [RC = 5]
545E	Missing operand(s) [RC = 5]
566E	Logical screen (*sl1,sw1,sh1,sv1*) is outside the virtual screen [RC = 5]
567E	Logical screens (*sl1,sw1,sh1,sv1*) and (*sl2,sw2,sh2,sv2*) overlap each other [RC = 5]
697E	The logical screens must cover the full virtual screen width [RC = 5]

Return Codes

0	Normal
1	The total number of lines or columns for the multiple logical screens exceeds the virtual screen size
3	Operand is only valid for display terminal
4	Each logical screen must contain at least 5 lines and 20 columns, or too many logical screens defined
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

```
set screen 2
```

splits the screen horizontally into two logical screens.

```
set screen 2 V
```

splits the screen vertically into two logical screens.

```
set screen size 14 10
```

splits the screen horizontally into two logical screens, the first of which is 14 lines long and the second of which is 10 lines long.

```
set screen width 25 25 30
```

creates three logical vertical screens, 25, 25, and 30 columns wide, respectively.

```
set scr def 16 40 1 1 16 40 1 41 8 80 17 1
```

results in the following screen layout on a 24 x 80 virtual screen.

```
    (1,1)                  (1,41)
      *************************************************
      *                       *                       *
      *                       *                       *
      *                       *                       *
      *        16 x 40        *        16 x 40        *
      *                       *                       *
      *                       *                       *
      *                       *                       *
(17,1)*************************************************
      *                                               *
      *                                               *
      *                    8 x 80                     *
      *                                               *
      *                                               *
      *************************************************
```

SET SELECT

Use the SELECT option to designate a "selection level" for specified lines. A selection level is a positive value assigned to a line in a file. Various lines in a file may be grouped logically by assigning them the same selection level. This subcommand can be used effectively with the SET DISPLAY, SET SHADOW, and SET SCOPE subcommands.

Format

[SET]	**SELect** [+ | -]*n* [*target* 1]

Operands

n

is the selection level for the lines specified. "n" is a number and specified by itself without a + or − assigns an absolute "n" value as the selection level for the lines specified.

+|-

adds(+) or subtracts(−) "n" from the current selection level(s) of the lines specified. For example, if you used "SET SELECT 8 3" the selection level "8" would be assigned to three lines in the file. To assign a level of "6" to the three lines you have just assigned a selection level of "8" to, use "SET SELECT −2 3" or "SET SELECT 6 3." Likewise, to assign a level of "10" to the three lines you have just assigned a selection level of "8" to you could either use "SET SELECT +2 3" or "SET SELECT 10 3."

target

defines the number of lines to be assigned a selection level. The selection level is assigned from the current line up to, but not including, the specified target line. If you enter an asterisk (*), the selection level is assigned to the rest of the file. If you omit the target, only the current line's selection level is set.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Initial Setting

`SELECT 0` (for the entire file)

Usage Notes

1. Selective line editing can be used in a macro to control both the action of the editor and the screen display. It consists of four subcommands, SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW, which work together in the following manner. You can use SET SELECT to assign a "selection level," or value, to one or more lines in a file. Lines can be logically grouped by assigning them the same selection level. SET DISPLAY may be used in conjunction with SET SELECT to display those lines which have the same selection level. SET SCOPE defines the set of lines that the editor can act upon. SCOPE DISPLAY is the initial setting and, as such, restricts editor action to only those lines that are defined by SET DISPLAY. By default, SET SHADOW displays a notice

indicating how many lines are not being displayed in the physical position of the excluded lines in the file. If SET SHADOW is "OFF," only those lines defined by SET DISPLAY will appear on the screen, with no shadow lines to indicate where lines are not being displayed.

2. For examples of this and other selective line editing subcommands, you can examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display.
3. If you specify −n, which would cause the selection level of the line to be negative, the selection level is set to zero instead.
4. TOF and EOF will always be set to the n1 value of SET DISPLAY, even though you cannot assign a selection level to either of them.
5. For more information on selective line editing, see Appendix B, "Effects of Selective Line Editing Subcommands" on page 405.

Responses

If you specify that SET SELECT is to occur on multiple lines and it does occur, the current line pointer will be:

a. Unchanged if SET STAY ON has been issued
b. Moved to the last line assigned a selection level, if SET STAY OFF is in effect (the default).

If SCOPE DISPLAY is set and the current line's selection level is not within the SET DISPLAY n1 n2 values, the editor forces the current line out of the display, causing the first line following it in the scope to be the new current line. This can occur if:

a. You issue SET SELECT changing the current line's selection level
b. You issue SET DISPLAY, changing lines that are displayed
c. You issue SET SCOPE DISPLAY and the previous setting was SCOPE ALL.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
1	TOF or EOF reached
2	Target not found
4	No change occurred
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

Figure 14 demonstrates the usage of the selective line editing subcommands: SET SELECT, SET SHADOW, SET DISPLAY, AND SET SCOPE.

Elbert had a record collection for which he established a "names" file. He used the following organization scheme:

```
NICK      O for opera
          C for classical

COMPOSER  Name of Composer

NAME      Name of Composition

ADDRESS   C or O, depending on type,
          followed by the assigned number
          on the record jacket.
```

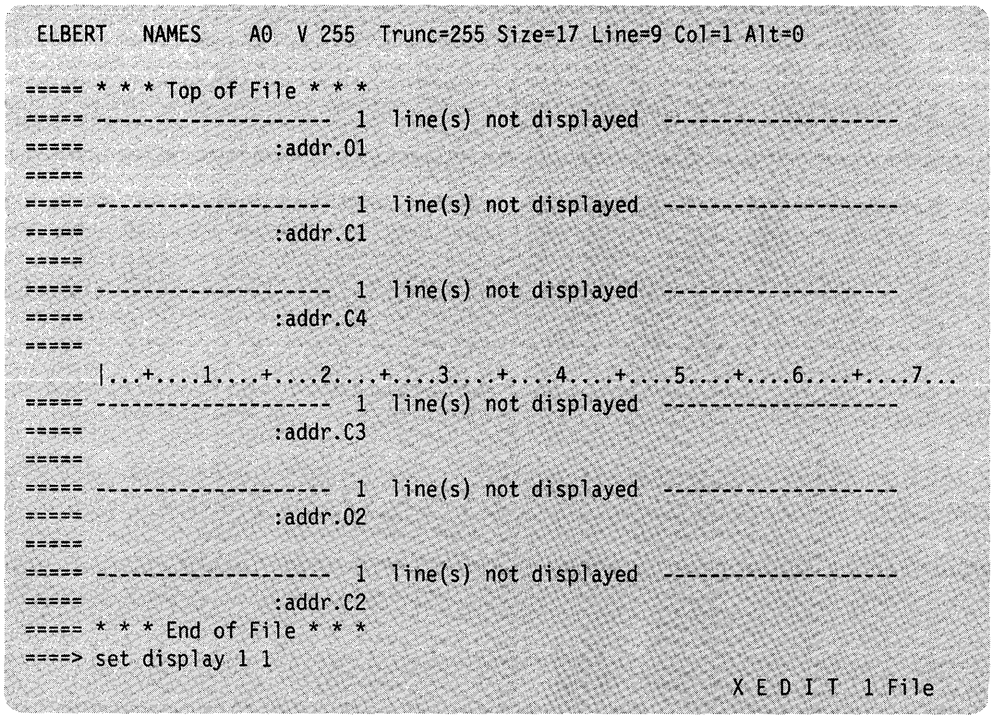
Elbert wanted to be able to list all the records in his collection by type. Using the macro below, Elbert located all the lines in the file which contained :nick.O and assigned a selection level of 1 to them. He also located all lines in the file which contained :nick.C, assigning a selection level of 2 to them.

```
/**** Xedit macro to set selection levels for Elbert's Records ...******/
'COMMAND TOP'                    /* Start at the Top of the file      */
'COMMAND SET WRAP OFF'           /* Do not wrap on Locate command!    */
'COMMAND SET MSGMODE OFF'        /* Suppress 'TARGET NOT FOUND' msg   */
'COMMAND SET CASE M I'           /* Ignore case on searches           */
'COMMAND LOCATE /:NICK.O/'       /* Look for first :nick.O            */
Do until rc ¬= 0                 /* Loop until LOCATE fails (rc > 0)  */
  'COMMAND SET SELECT 1 1'       /* Set selection level 1             */
  'COMMAND LOCATE /:NICK.O/'     /* Now, find the next one...         */
  End                            /* End of loop                       */
'COMMAND TOP'                    /* Start at top again                */
'COMMAND LOCATE /:NICK.C/'       /* Now look for :nick.C              */
Do until rc ¬= 0                 /* Loop until LOCATE  fails (rc>0)   */
  'COMMAND SET SELECT 2 1'       /* Set selection level               */
  'COMMAND LOCATE /:NICK.C/'     /* Find next occurrence              */
  End                            /* End of Loop                       */
'COMMAND TOP'                    /* Go back to the top                */
'COMMAND SET MSGMODE ON'         /* Now we want messages again        */
Exit                             /* Exit the macro                    */
```

Note: Please note that after the macro executes, you will be at the top of your file. For purposes of allowing you to see more of the file, however, we have moved down in the file to make line 9 the current line.

Figure 14 (Part 1 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

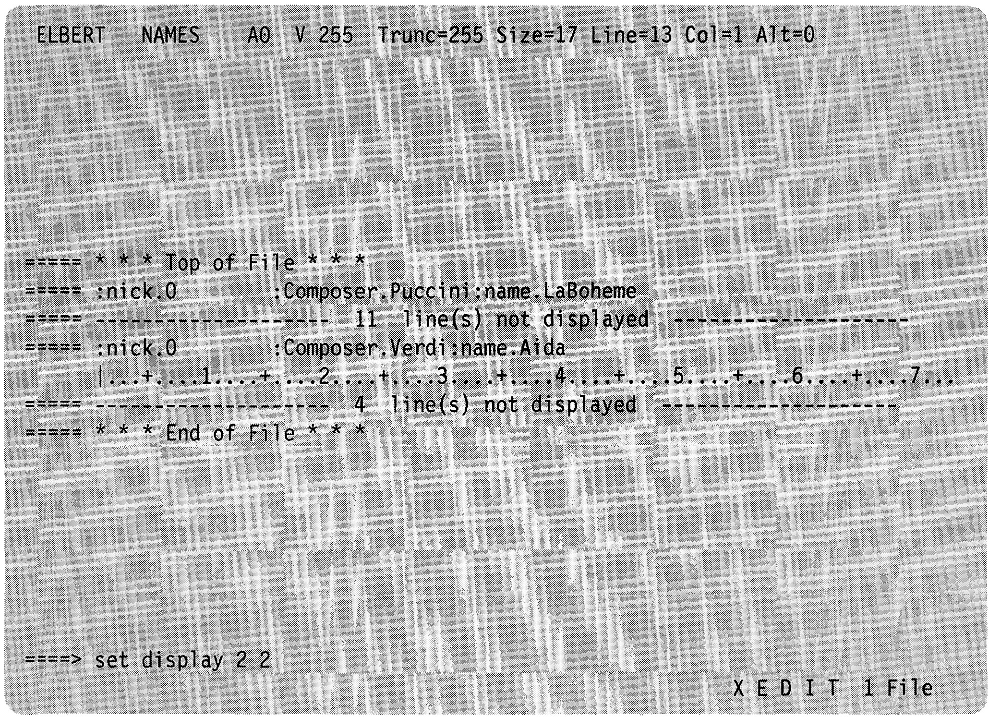
All nick.O or nick.C lines in the file have now been assigned a selection level. They appear as shadow lines, because the initial setting of SET DISPLAY is 0 0 and any lines in the file not assigned a selection level of 1 or 2 will have a selection level of 0. In order to look at all the nick.O compositions, Elbert issued the command, SET DISPLAY 1 1 as shown below.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=9 Col=1 Alt=0

===== * * * Top of File * * *
===== -------------------- 1  line(s) not displayed  --------------------
=====                :addr.O1
=====
===== -------------------- 1  line(s) not displayed  --------------------
=====                :addr.C1
=====
===== -------------------- 1  line(s) not displayed  --------------------
=====                :addr.C4
=====
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== -------------------- 1  line(s) not displayed  --------------------
=====                :addr.C3
=====
===== -------------------- 1  line(s) not displayed  --------------------
=====                :addr.O2
=====
===== -------------------- 1  line(s) not displayed  --------------------
=====                :addr.C2
===== * * * End of File * * *
====> set display 1 1
                                                            X E D I T  1 File
```

Figure 14 (Part 2 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

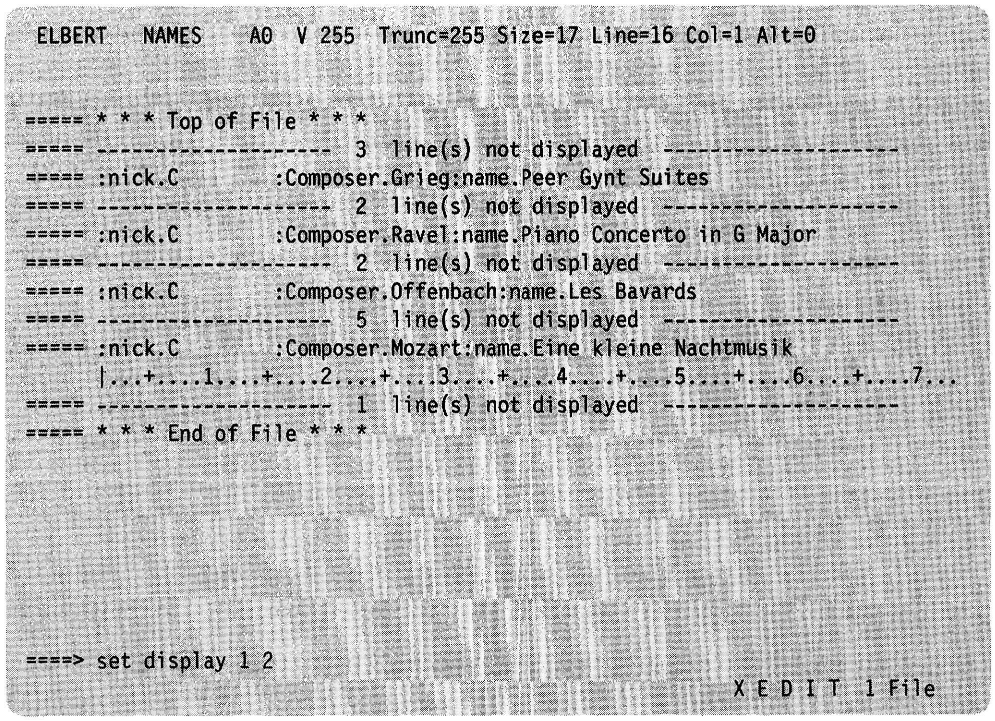
The resulting display listed only those lines in the file with a selection level of 1 or those assigned to nick.O lines in the file. Elbert then set the display to list all lines assigned a selection level of 2, by issuing the subcommand below.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=13 Col=1 Alt=0









===== * * * Top of File * * *
===== :nick.0        :Composer.Puccini:name.LaBoheme
===== --------------------  11  line(s) not displayed  --------------------
===== :nick.0        :Composer.Verdi:name.Aida
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== --------------------  4  line(s) not displayed  --------------------
===== * * * End of File * * *








====> set display 2 2
                                                              X E D I T  1 File
```

Figure 14 (Part 3 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

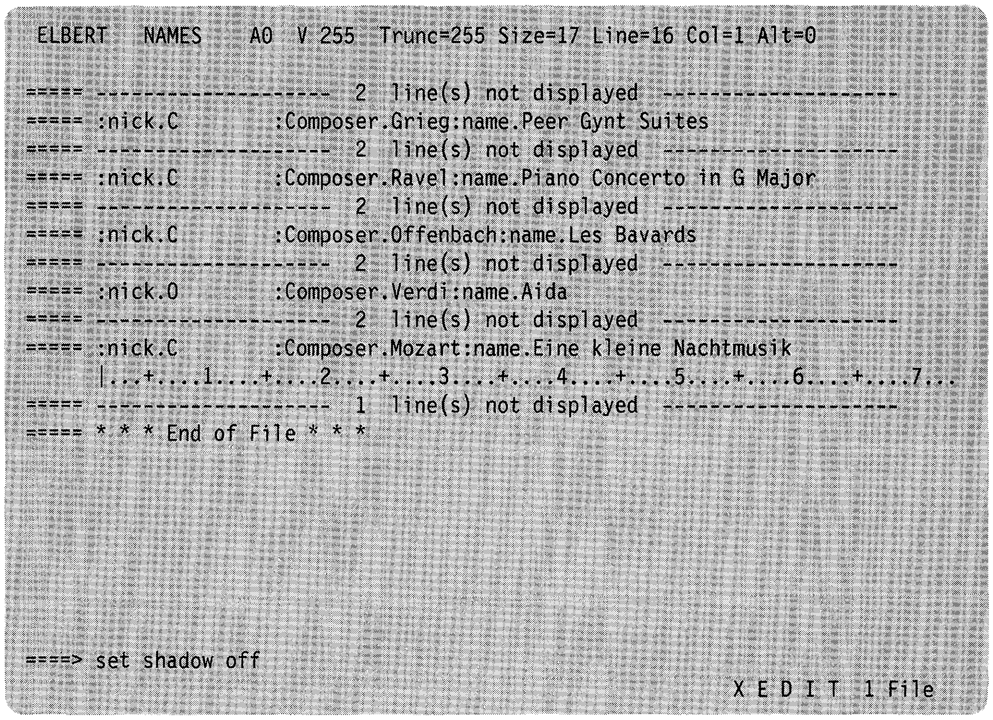
The display now listed only Elbert's classical selections (nick.C). To list both types of selections, Elbert issued the command, SET DISPLAY 1 2. That displayed all lines in the file with either a selection level of 1 or 2.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0


===== * * * Top of File * * *
===== -------------------- 3  line(s) not displayed --------------------
===== :nick.C        :Composer.Grieg:name.Peer Gynt Suites
===== -------------------- 2  line(s) not displayed --------------------
===== :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
===== -------------------- 2  line(s) not displayed --------------------
===== :nick.C        :Composer.Offenbach:name.Les Bavards
===== -------------------- 5  line(s) not displayed --------------------
===== :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== -------------------- 1  line(s) not displayed --------------------
===== * * * End of File * * *








====> set display 1 2
                                                          X E D I T  1 File
```

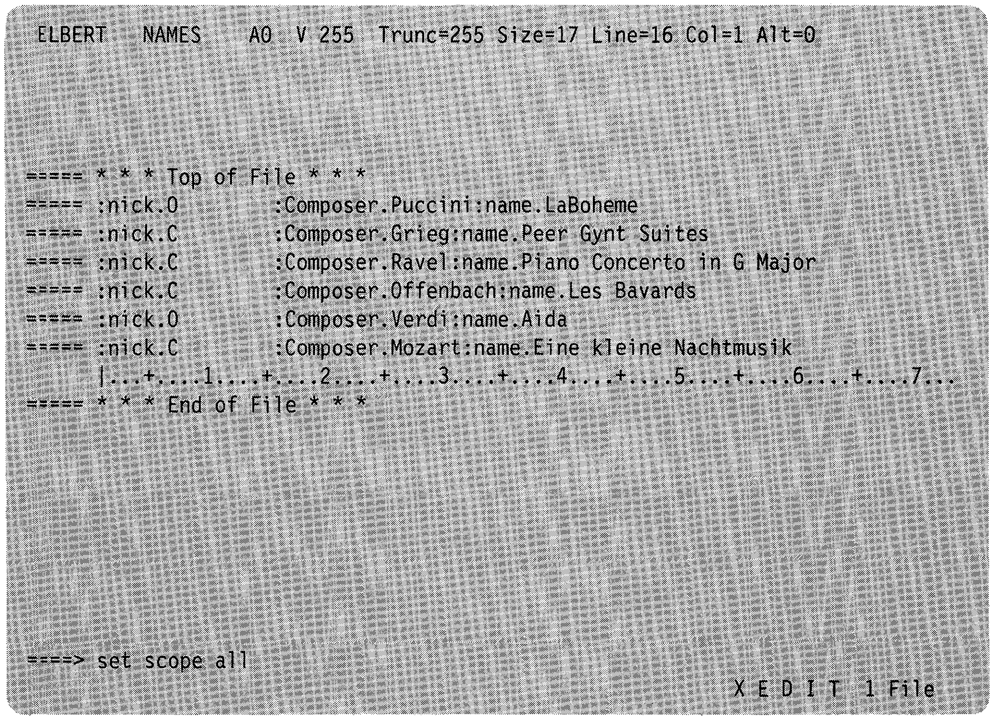
Figure 14 (Part 4 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

Annoyed by the appearance of shadow lines in the screen display, Elbert issued the subcommand, SET SHADOW OFF to eliminate shadow lines from the screen display.



```
 ELBERT   NAMES     A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0

===== --------------------  2  line(s) not displayed  -------------------
===== :nick.C        :Composer.Grieg:name.Peer Gynt Suites
===== --------------------  2  line(s) not displayed  -------------------
===== :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
===== --------------------  2  line(s) not displayed  -------------------
===== :nick.C        :Composer.Offenbach:name.Les Bavards
===== --------------------  2  line(s) not displayed  -------------------
===== :nick.O        :Composer.Verdi:name.Aida
===== --------------------  2  line(s) not displayed  -------------------
===== :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== --------------------  1  line(s) not displayed  -------------------
===== * * * End of File * * *







====> set shadow off
                                                           X E D I T  1 File
```

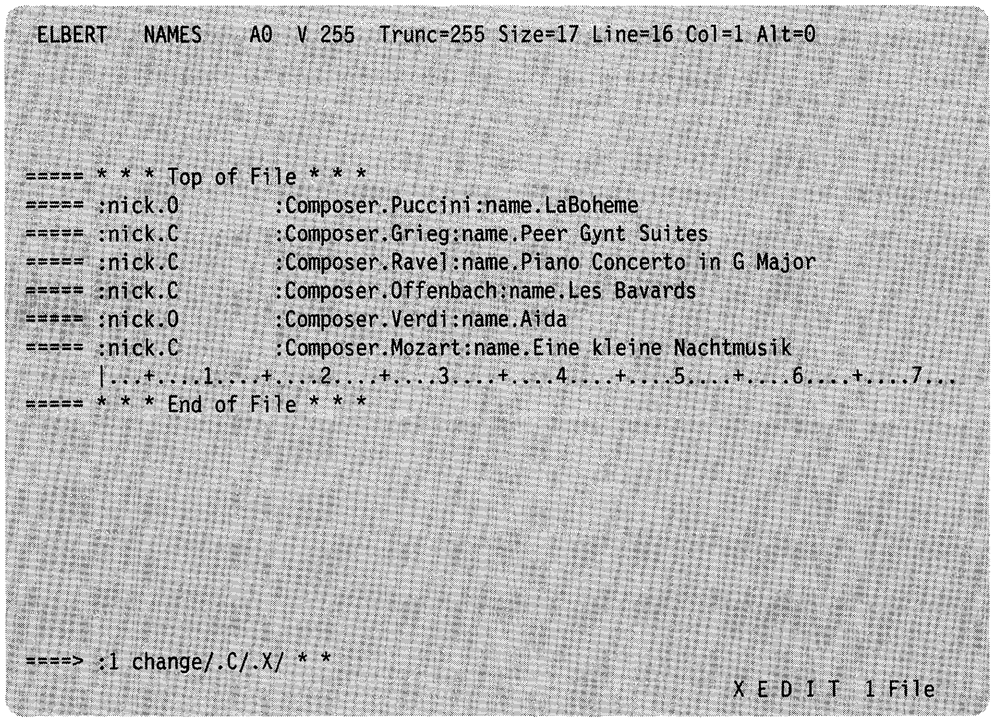
The result is a listing of all lines in the file which have been assigned selection levels. Please note the absence of shadow lines.



```
 ELBERT   NAMES     A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0



===== * * * Top of File * * *
===== :nick.O        :Composer.Puccini:name.LaBoheme
===== :nick.C        :Composer.Grieg:name.Peer Gynt Suites
===== :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
===== :nick.C        :Composer.Offenbach:name.Les Bavards
===== :nick.O        :Composer.Verdi:name.Aida
===== :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== * * * End of File * * *









====> set scope all
                                                           X E D I T  1 File
```

Figure 14 (Part 5 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

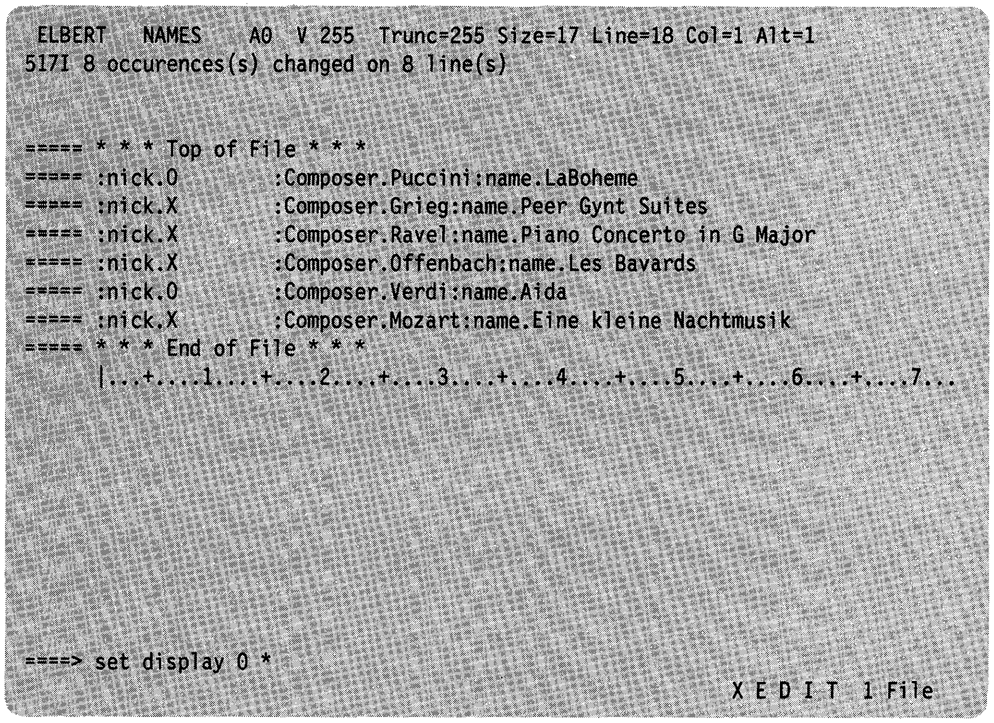
Elbert decided to add some country and western records to his collection. In order to avoid confusion between his classical records and his country records, he changed .C to .X. In the previous screen, he issued SET SCOPE ALL to expand editor control to include the entire file. This was necessary so that both the lines listing composers' names and those listing record addresses would be changed.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0




===== * * * Top of File * * *
===== :nick.O        :Composer.Puccini:name.LaBoheme
===== :nick.C        :Composer.Grieg:name.Peer Gynt Suites
===== :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
===== :nick.C        :Composer.Offenbach:name.Les Bavards
===== :nick.O        :Composer.Verdi:name.Aida
===== :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== * * * End of File * * *








====> :1 change/.C/.X/ * *
                                                            X E D I T  1 File
```

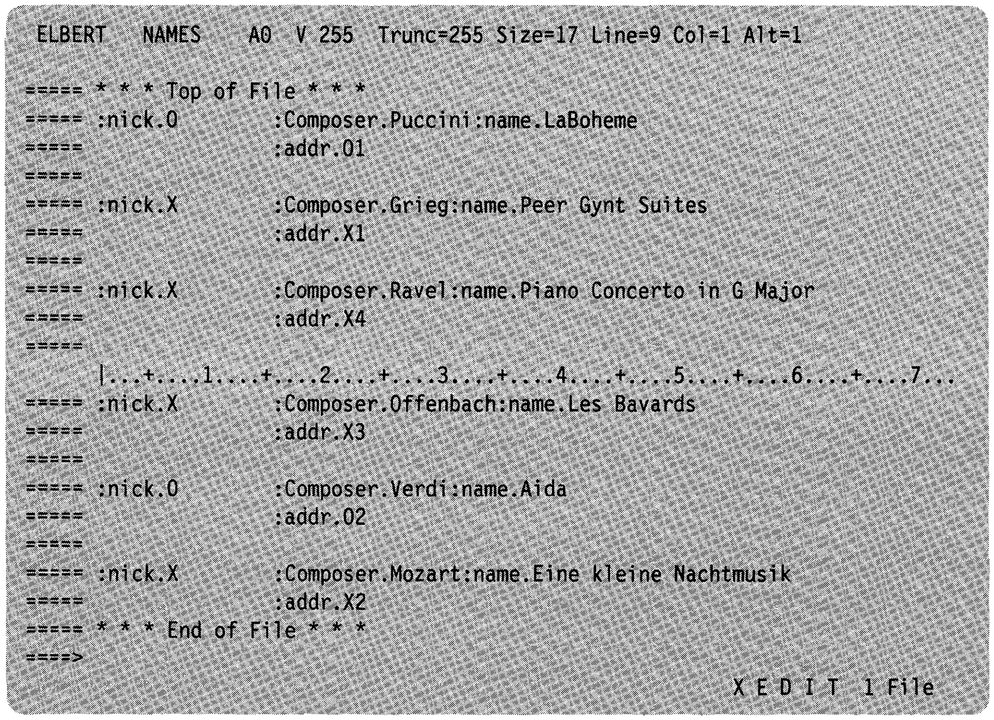
Figure 14 (Part 6 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

In order to display the entire file, Elbert issued SET DISPLAY 0 *. This displayed all selection levels of lines, starting with 0.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=18 Col=1 Alt=1
5171 8 occurences(s) changed on 8 line(s)


===== * * * Top of File * * *
===== :nick.O        :Composer.Puccini:name.LaBoheme
===== :nick.X        :Composer.Grieg:name.Peer Gynt Suites
===== :nick.X        :Composer.Ravel:name.Piano Concerto in G Major
===== :nick.X        :Composer.Offenbach:name.Les Bavards
===== :nick.O        :Composer.Verdi:name.Aida
===== :nick.X        :Composer.Mozart:name.Eine kleine Nachtmusik
===== * * * End of File * * *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...










====> set display 0 *
                                                        X E D I T  1 File
```

Following the changes, Elbert's file looks like the display below. Please note that we have changed the current line, without issuing the subcommand here, so that you may see more of the file.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=9 Col=1 Alt=1

===== * * * Top of File * * *
===== :nick.O        :Composer.Puccini:name.LaBoheme
=====                :addr.O1
=====
===== :nick.X        :Composer.Grieg:name.Peer Gynt Suites
=====                :addr.X1
=====
===== :nick.X        :Composer.Ravel:name.Piano Concerto in G Major
=====                :addr.X4
=====
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== :nick.X        :Composer.Offenbach:name.Les Bavards
=====                :addr.X3
=====
===== :nick.O        :Composer.Verdi:name.Aida
=====                :addr.O2
=====
===== :nick.X        :Composer.Mozart:name.Eine kleine Nachtmusik
=====                :addr.X2
===== * * * End of File * * *
====>
                                                        X E D I T  1 File
```

Figure 14 (Part 7 of 7). SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE

SET SERIAL

Use the SERIAL option to control file serialization.

Format

```
[SET]     SERial   ON        [incrno   [startno]]
                             [10       [10     ]]

                   ALL       [incrno   [startno]]
                             [1000     [1000   ]]

                   string    [incrno   [startno]]
                             [10       [10     ]]

                   OFF
```

Operands

ON
adds a serial identification in the last eight columns of each file line. The serial identification consists of the first three letters of the file name plus five digits, where startno is the starting value and incrno is the increment.

ALL
adds a serial identification in the last eight columns of each file line. The serial identification consists of eight digits, where startno is the starting value and incrno is the increment.

string
adds a serial identification in the last eight columns of each file line. The serial identification consists of the characters specified in string, and, if string contains fewer than eight characters, a number, where startno is the starting value and incrno is the increment. If string contains more than eight characters, it is truncated to eight characters.

OFF
specifies that the serial identification area is not to be updated the next time the file is written to disk or directory.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Notes

1. Only files with a fixed record format can be serialized. Also, the room between TRUNC and LRECL must be at least eight characters in order to serialize the file (if not, message 560E is displayed).
2. The serialization takes place only when a FILE or SAVE subcommand is issued.
3. To remove the serial identification from a file whose logical record length is 80, you can use the following sequence of subcommands:

```
set trunc 80
set zone 1 80
set serial off
top
next
clocate :73
cdelete 8
repeat *
```

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
558E	Wrong file format for serialization [RC=5]
560E	Not enough space for serialization between TRUNC and LRECL

Return Codes

0	Normal
5	Missing or invalid operand or number, or wrong file format
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SHADOW

Use the SHADOW option to display a notice (called a shadow line) that indicates how many lines have been excluded from the display. The shadow line appears in the physical position in the file of the excluded lines(s). For an illustration of how SET SHADOW can be used effectively in combination with the other selective line editing subcommands (SET SELECT, SET SCOPE, and SET DISPLAY), see SET SELECT in this publication.

Format

[SET]	**SHADow**	**ON** **OFF**

Operands

ON
displays a shadow line which cites the number of lines omitted in the display. Each shadow line is in the physical position of the omitted line or block of lines.

OFF
causes lines not displayed by virtue of the SET DISPLAY subcommand to disappear totally from the screen, thus allowing no visual representation of them.

Initial Setting

```
SHADOW ON
```

Usage Notes

1. For examples of this and other selective line editing subcommands, you can examine the following IBM-supplied macros: ALL, (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display. See also Appendix B, "Effects of Selective Line Editing Subcommands" on page 405.
2. If a prefix macro is entered on a shadow line, then the line number of the first excluded line is passed to the macro as "pline" (the line number on which the prefix macro was entered). (Refer to SET PENDING in this publication.)
3. With SET NUMBER ON, the sequence numbers enable you to determine which lines are excluded from the display. This is especially useful if SHADOW OFF has been set. If SHADOW is ON, the sequence number in a shadow line is the first line in a block of excluded lines.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SIDCODE

Use the SIDCODE option to insert a character string in every line of an update file. The string is inserted in the first eight columns of the last 17 columns of the file line (*lrecl*—16 to *lrecl*—9). For example, if you have a file with fixed, eighty-character lines, the editor inserts the string in columns 64-71. If the string is less than eight characters, it is padded on the right with blanks. Any data in the eight columns is overlaid.

Format

[SET]	**SIDcode** [*string*]

Operand

string
is the character string (1-8 characters) that is to be inserted in every line of an update file. If you omit string, SIDCODE is set to blanks. If string contains more than eight characters, it is truncated to eight characters.

Initial Setting

SIDCODE is set to blanks (or as specified in the XEDIT command or the LOAD subcommand).

Usage Notes

1. The string is inserted in every line of an update file when at least one change is made to the update file and it is filed (or saved).
2. Data will not be overlaid in lines with a character in column *lrecl*—8.
3. See also the SIDCODE option of the XEDIT command.
4. When SIDCODE is set to all blanks (the default setting), columns *lrecl*—16 to *lrecl*—9 are left unchanged.

Messages

520E	Invalid operand: *operand* [RC = 5]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SPAN

Use the SPAN option to specify if a character string, which is the subject of a target search, must be included in one line in order to be found, or if it may span a specified number of lines.

Format

```
[SET]    SPAN    ON    [Blank     [n]]
                       [Noblank   [*]]

                 OFF
```

Operands

ON
specifies that lines are concatenated during a search for a character string. Trailing blanks are removed.

Blank
specifies that one blank character is inserted between consecutive file lines and must be considered when defining the target. This is the standard setup for SCRIPT and other text files. Lines are temporarily concatenated for the search, separated by one blank. Any trailing blanks are ignored.

Noblank
specifies that consecutive file lines are concatenated temporarily but are not separated by a blank.

n
specifies the number of consecutive file lines that a character string can span. If an asterisk (*) is specified the rest of the file is searched.

OFF
specifies that a character string must be included within one file line in order to match a target.

Initial Setting

```
SPAN  OFF  BLANK  2
```

Usage Notes

1. When consecutive file lines are searched for a string, the search occurs within the columns defined in the SET ZONE subcommand. Portions of consecutive lines are concatenated for the search, separated or not by blanks (as specified in SET SPAN ON BLANK or SET SPAN ON NOBLANK).

2. In SCRIPT or other text processing files, SET SPAN ON BLANK and SET VARBLANK ON can be combined to advantage.

   If the file contains

left zone	**right zone**
1	**72**

```
                   The house
          was near
```

on two consecutive lines, the two file lines, when concatenated for the search if SET SPAN is ON, would have many blanks (depending on the logical record length) between "house" and "was," in addition to the single blank inserted because of SET SPAN ON BLANK. However, the SET VARBLANK ON would still permit the target /house was/ to match the text.

On the other hand, a programmer editing an object deck produced by a compiler (file type TEXT, zones 1 72) or a PL/I source program (file type PLI, zones 2 72) should use SET SPAN ON NOBLANK, since no implicit blanks are assumed to separate lines.

For example:

left zone	**right zone**
1	**72**

```
                                        X=' THE LIT
TLE HOUSE';
```

The target `/THE LITTLE HOUSE/` will be found.

Messages

520E	Invalid operand: *operand* [RC = 5]
543E	Invalid number: *number* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SPILL

Use the SPILL option to specify if data is spilled onto new lines or lines are truncated following these subcommmands: CHANGE, CINSERT, COVERLAY, CREPLACE, EXPAND, GET, INPUT, MERGE, OVERLAY, REPLACE, SHIFT, (and macros that use these subcommands internally, including CAPPEND, JOIN and PRFSHIFT (>, > >)).

Format

[SET]	SPILL ON OFF WORD

Operands

ON
specifies that characters pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character that would have gone beyond the truncation column.

OFF
specifies that any characters pushed beyond the truncation column are lost.

WORD
specifies that characters pushed beyond the truncation column are used to create one or more new lines, as necessary. However, a word is not split between lines if possible. It is moved, or "spilled," in its entirety. Thus, the first character of each new line is the first non-blank character following the last blank within the truncation setting (see SET TRUNC) on the affected line.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Notes

1. SET SPILL affects only SHIFT issued with the RIGHT operand; any characters shifted past the truncation column spill onto new lines. SET SPILL has no effect on SHIFT LEFT; data shifted to the left past the zone1 column is lost.
2. For GET, the LRECL is used for the truncation column instead of the TRUNC setting.
3. For JOIN, SPILL OFF functions like SPILL ON. JOIN does not truncate data.
4. Recovered lines (see RECOVER) are not spilled.
5. New lines which are inserted because of the SPILL setting will always be inserted starting at column 1, regardless of whether SET IMAGE is ON or OFF.
6. When SET VERIFY is ON and a line is changed by a subcommand and spilled as a result, the last line inserted due to SET SPILL will be displayed.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

```
With SET SPILL WORD:

===== Now is the time for all good men to come to the aid of their party.

====> c/good/good, courageous, stout-hearted/

results in:

===== Now is the time for all good, courageous, stout-hearted men to come to
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7.T.
===== the aid of their party.

With SET SPILL ON, the same CHANGE subcommand results in:

===== Now is the time for all good, courageous, stout-hearted men to come to t
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7.T.
===== he aid of their party.
```

SET STAY

Use the STAY option to specify whether or not the line pointer is to move when a string that is the object of a LOCATE target search, FIND, FINDUP, NFIND, or NFINDUP is not found. Also, use the STAY option to specify whether or not the line pointer is to move when a string that is the object of the target search for CHANGE, COUNT, COMPRESS, EXPAND, LOWERCAS, SET SELECT, SHIFT, or UPPERCAS is found.

Format

[SET]	**STAY ON** **OFF**

Operands

ON
specifies that the line pointer is *not to move.*

OFF
specifies that the line pointer moves.

Initial Setting

```
STAY OFF
```

Usage Notes

1. SET STAY applies to LOCATE target searches and FIND family searches issued to the end of file (or end of range). With SET STAY OFF, the null End of File (or End of Range) line will become the new current line if a search is unsuccessful. The Top of File (or Top of Range) line becomes the new current line if the search was in a backward direction. With SET STAY ON, the current line remains the same.
2. SET STAY applies also to the following subcommands: CHANGE, COUNT, COMPRESS, EXPAND, LOWERCAS, SET SELECT, SHIFT, and UPPERCAS. With SET STAY OFF, the last line examined or acted upon becomes the new current line; with SET STAY ON, the line pointer does not move when the subcommand is executed.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET STREAM

Use the STREAM option to specify whether the editor is to search the entire file or only the current line for a character string that is a column-target in a CLOCATE or CDELETE subcommand.

Format

[SET]	**STReam** **ON** **OFF**

Operands

ON
specifies that the editor begins searching at the character following the column pointer and continues to the bottom of the file (or of the range). (If the search is in the other direction, the editor begins searching at the character preceding the column pointer and continues to the top of the file (or of the range)).

OFF
specifies that the editor searches only the current line (within the limits defined by the SET ZONE subcommand).

Initial Setting

```
STREAM  ON
```

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SYNONYM

The SET SYNONYM subcommand has three formats.

Use the first format to specify whether or not the editor is to look for synonyms.

Use the second format to assign a synonym to any existing subcommand or macro (except prefix subcommands or prefix macros) and, optionally, to define an abbreviation for the synonym. (You must use the SET PREFIX subcommand to define a synonym for a prefix subcommand or macro.)

The third format provides a more complex function, which is used when the synonym represents a subcommand whose operands are entered in a different order than the XEDIT subcommand operands. The operands are automatically rearranged into the order expected by XEDIT.

Format

[SET]	**SYNonym** **ON** **OFF**
	SYNonym [**LINEND** *char*] *newname* [*n*] *oldname*
	SYNonym [**LINEND** *char*] *newname* [*n* [*format* **1...** *format n*]] *oldname* [**&1...&***n*]

Operands

ON
specifies that the editor is to look for synonyms.

OFF
specifies that the editor is not to look for synonyms.

LINEND *char*
specifies a character that is interpreted as a line end character regardless of the current SET LINEND setting at the time the synonym is used.

newname
is the synonym to be assigned to the subcommand or macro. The synonym can be an alphabetic string from one to eight characters long, or it can be a single alphabetic, numeric, or special character.

n
is the minimum number of characters that must be entered for the synonym to be accepted, that is, its minimum abbreviation. If you omit n, the full synonym name (newname) must be entered.

format1...formatn
specifies the format for the "new" operands. Each operand must be entered in the format specified. Format is defined as one of the following:

&	the operand is delimited by blanks.
&/	the operand is a string enclosed by delimiters, for example, /ABC/.

&. the operand is the first of two strings that are separated by a common delimiter. The second string would be specified as &/.

&* represents all the remaining data.

oldname
is the name of a subcommand or macro for which you are creating a synonym. It may be a compounded name (for example, QUERY PF) or a subcommand name followed by its arguments.

&1....&n
specifies the relative order in which the new operands are to be inserted in the XEDIT subcommand, even though they are entered in a different order with the synonym. **&1** represents the first operand in the synonym operand list, **&2** represents the second operand, and so forth. The list specified here is positional and determines how the operands are to be rearranged.

Initial Setting

SYNONYM ON and the following synonyms are defined:

```
SET SYNONYM ALTER       2 ALTER
SET SYNONYM CAPPEND     2 CAPPEND
SET SYNONYM FILE        4 COMMAND PFILE
SET SYNONYM SSAVE       2 COMMAND SAVE
SET SYNONYM FFILE       2 COMMAND FILE
SET SYNONYM HELP        1 HELP
SET SYNONYM HEXTYPE     4 HEXTYPE
SET SYNONYM JOIN        1 JOIN
SET SYNONYM MODIFY      3 MODIFY
SET SYNONYM QUIT        4 COMMAND PQUIT
SET SYNONYM QQUIT       2 COMMAND QUIT
SET SYNONYM SAVE        4 COMMAND PSAVE
SET SYNONYM SPLIT       2 SPLIT
SET SYNONYM STATUS      4 STATUS.
```

Usage Notes

1. The "newname" operand can be the name of an existing XEDIT subcommand. In this case, the SYNONYM subcommand defines a new meaning for that subcommand name. The original meaning can be obtained by using:

   command *oldname* ...

   or

   set synonym off
   oldname...

2. Newname can be alphabetic or it can be a single special character. For example:

   ```
   syn / 1 clocate/
   ```

   causes implicit LOCATEs, such as a /string/ target, to become CLOCATEs.

3. A synonym should not be defined for a name that is already defined as a synonym. For example:

   ```
   synonym erase delete
   synonym remove erase
   ```

   If REMOVE is entered, the editor looks for a subcommand called ERASE, not for a subcommand called DELETE.

```
synonym add delete
synonym linend $ SXMS locate/A/$ADD
```

If SXMS is entered, a line is added after the line containing string A. The DELETE does not occur.

Messages

497E	Minimum abbreviation is between SO and SI [RC=5]
520E	Invalid operand: *operand*
544E	Invalid hex data: *xxxxxxxx* [RC=5]
545E	Missing operand(s) [RC=5]
547E	Synonym definition incomplete [RC=5]
548E	Invalid synonym operand: *operand* [RC=5]
549E	Synonym abbreviation too large [RC=5]
550E	Too many operands in synonym definition [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Simple form:

1. `syn down 1 up`
2. `syn qpf query pf`
3. `syn linend | QK 2 query pf|query pa|query enter`

   Entering QK displays the PF, PA, and ENTER key settings.

Complex form:

1. `syn putfile 3 & & & & put &4 &1 &2 &3`

   If you enter the command as follows:

   put *fn ft fm n*

   The editor rearranges it as follows:

   put *n fn ft fm*

2. `syn alter 2 &. &/ &* change /X'&1'/X'&2'/&3`

   This example translates the subcommand:

   `alter /7B/15/ * *`

   to the following:

   `change /X'7B'/X'15'/ * *`

SET TABLINE

Use the TABLINE option to display, on a specified line, a "T" in every tab column according to the current tab settings (See SET TABS in this book).

Format

[SET]	**TABLine** **ON** [M[+*n* |-*n*] | [±|-]*n*] **OFF**

Operands

ON
displays the tab line.

M [+*n*|-*n*]
specifies the line in which the tab line is displayed. M stands for "middle of the screen" (rounded up for odd-sized screens). M may be combined with a constant positive or negative integer to mean n lines below the middle of the screen (M + n), or n lines above the middle of the screen (M − n).

[+|-]*n*
specifies the line in which the tab line is displayed. n or + n (+ is implicit) specifies that the tab line is displayed n lines from the top of the screen: − n specifies that the tab line is displayed n lines from the bottom of the screen.

OFF
removes the tab line from the screen.

Initial Setting

`TABLINE OFF -3` (three lines from the bottom of the screen).

Usage Notes

1. TABLINE can be set on the same line as the scale line (see SET SCALE).
2. If a line occupies more than one screen line and the tab line is set on an adjacent line, the tab line is not displayed. The tab line is redisplayed when the line pointer moves to a file line that occupies only one screen line.

Response

The specified line contains a "T" in every tab column. For example:

```
T  T  T  T  T  T  T  T
```

Messages

520E	Invalid operand: *operand* [RC = 5]
521E	Invalid line number [RC = 5]
526E	Option *option* valid in display mode only [RC = 3]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
3	Operand is valid only for display terminal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

tabline on m	displays the tab line in the middle of the screen.
tabline on m+3	displays the tab line three lines below the middle of the screen.
tabline on −3	displays the tab line three lines from the bottom of the screen.

SET TABS

Use the TABS option to define the logical tab stops for a file.

Format

[SET]	TABS *n*1 [*n*2... *n*28]

Operand

n1...n28
define the column numbers for logical tab settings. You may specify up to 28 numbers, separated from each other by at least one blank.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Notes

1. A SET IMAGE ON subcommand must be in effect for a X'05' to be recognized as a tab character in an input line.
2. A line containing tab characters can be entered from the terminal or the console stack. A tab character in an input line causes "space" characters to be inserted up to (but not including) the next tab position. The "space" character is, in fact, the character defined by SET FILLER, whose default is a blank.
3. The COMPRESS subcommand inserts tab characters in a line and can be used with the EXPAND subcommand to realign data according to a SET TABS subcommand. (See "COMPRESS, EXPAND.")
4. On a display terminal, the SET TABS subcommand controls not only the *logical* tab settings but also the *physical* tab settings. A PF key can be set up to function like a tab key on a typewriter (see SET PFn TABKEY); each time the PF key is pressed, the cursor moves to the next column as defined in the SET TABS subcommand.
5. The default tab settings differ according to file type and can be displayed by QUERY TABS.
6. To define a tabulation character, issue the CMS command, SET INPUT. For example:

   ```
   cms set input > 05
   ```

   defines a " > " as the tabulation character.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]
575E	Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} columns defined [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET TERMINAL

Use the TERMINAL option to specify whether a terminal is to be used in line mode or in full screen mode. (This option is meaningful only on a display terminal.)

Format

[SET]	**TERMinal**	**Typewriter** **Display**

Operands

Typewriter
specifies that the display terminal is to be used in line mode.

Display
specifies that the terminal is to be used in full screen mode.

Initial Setting

TERMINAL DISPLAY

Usage Notes

1. With a remote display terminal, full screen performance depends on the line transmission rate; if it is too slow, line mode (TYPEWRITER) may be specified.
2. In case of a severe transmission error while in full screen mode (DISPLAY), the editor automatically switches to line mode (TYPEWRITER).
3. If you are editing a file in full screen mode and you issue a DISCONN (disconnect) command, you must issue BEGIN after you reconnect, and then press ENTER to get the file image back on the screen.
4. If you are editing a file in line mode (SET TERMINAL TYPEWRITER), XEDIT does not recognize which key caused an attention interrupt. The definition which will be executed will be the CP definition of that key. To QUERY keys in line mode, use the CP QUERY PFnn command.

Messages

520E	Invalid operand: *operand* [RC=5]
526E	Option *option* valid in display mode only [RC=3]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
3	Operand is valid only for display terminal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET TEXT

Use the TEXT option to inform the editor and CMS if TEXT keys are available on the terminal.

Format

[SET]	**TEXT** **ON** **OFF**

Operands

ON
specifies that TEXT keys are available. You must issue a SET TEXT ON before using these keys so that proper character code conversion takes place.

OFF
specifies that no code conversion is to be performed for TEXT keys.

Initial Setting

```
Defined by the CMS setting.
```

Usage Notes

1. If a terminal is equipped with the TEXT feature, special TEXT keys are available. Before using these keys, you must inform the editor so that proper character code conversion takes place. There are two ways to inform the editor:
   a. Issue CMS SET TEXT ON.
   b. Issue the editor SET TEXT ON subcommand.
2. Because the conversion is costly, it is recommended that you issue the XEDIT subcommand SET TEXT OFF when you stop using the special keys.
3. Issuing SET TEXT ON while APL is ON causes APL to be set to OFF. Similarly, issuing SET APL ON while TEXT is ON causes TEXT to be set to OFF.
4. Changing the TEXT setting for XEDIT also changes the TEXT setting for CMS.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET TOFEOF

Use the TOFEOF option to control the display of the following notices:

Top of File
End of File
Top of Range
End of Range.

Format

[SET]	**TOFEOF**	**ON** **OFF**

Operands

ON
specifies that the notices listed above are displayed.

OFF
specifies that the notices listed above are not displayed.

Initial Setting

```
TOFEOF ON
```

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET TRANSLAT

Use the TRANSLAT option to control uppercase translation of specified characters. This option is designed to be used on terminals whose keyboards support characters other than English. By default, only English alphabetic characters are translated to uppercase.

Format

[SET]	**TRANSLat** *char* **1** *char* **2** [*char* **1** *char* **2**] ..... **OFF**

Operands

char1 char2
define a lowercase/uppercase pair of characters. Char1 is the lowercase character, and char2 is the uppercase character. Either may be specified as a single EBCDIC character or a two-digit hexadecimal character.

OFF
indicates that characters are not to be translated to uppercase.

Initial Setting

Only English characters are translated to uppercase.

Usage Notes

1. For data lines, translation occurs only if SET CASE UPPER is in effect or if a line(s) is changed to uppercase (by the UPPERCAS subcommand, for example). An existing data line is not translated to uppercase unless a change is made to the line, in which case the entire line is translated.
2. Commands are translated if SET CASE UPPER is in effect.
3. Translation specified by SET TRANSLAT occurs only for uppercase. For lowercase, the system-supplied translate table is used.

Messages

520E	Invalid operand: *operand* [RC = 5]
545E	Missing operand(s) [RC = 5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Assuming the terminal keyboard has the French character set and SET CASE UPPER:

```
set translat 51 E
```

```
(X'51' is "e" acute.)
```

User enters:

```
     / /    /          /
liberte egalite fraternite
```

which results in:

```
LIBERTE EGALITE FRATERNITE
```

SET TRUNC

Use the TRUNC option to define the truncation column, which is the last column in a line in which data may be entered or modified.

Format

[SET]	**TRunc** *n* *

Operand

n

specifies the column at which truncation (or "spilling" — see SET SPILL) occurs. If you specify an asterisk (*), the truncation column is set to the record length for the file type.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Notes

1. Data that is entered beyond the truncation column is not shifted due to character insertion or deletion.
2. When editing a file in update mode, you cannot SET TRUNC to a value greater than 72.

Messages

009E	Column *nn* exceeds record length (*nn*) [RC=5]
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
560E	Not enough space for serialization between TRUNC and LRECL

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET VARBLANK

Use the VARBLANK option to control whether or not the number of blank characters between two words is significant in a target search.

Format

[SET]	**VARblank**	**ON** **OFF**

Operands

ON
specifies that the number of blanks between two words can be variable and does not matter in searching for a target.

For example:

```
/the  house/
```

will match in the text:

```
"the       house"
```

and will also match

```
"the              house"
```

OFF
specifies that the number of blanks between two words is significant in a target search. Each blank in the target string matches one blank in the file.

Initial Setting

```
VARBLANK OFF
```

Usage Notes

1. SET VARBLANK ON is useful when editing text files, if periods at the end of sentences are not always followed by the usual two blanks or SCRIPT output files, where multiple blanks are generated between words for justification.
2. SET VARBLANK ON is useful with SET SPAN ON.

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET VERIFY

Use the VERIFY option:

- To control whether or not lines that are changed by subcommands are to be displayed (in the message area, for a display terminal)
- To define the columns that are to be displayed when a file appears on the screen. Optionally, data may be displayed in hexadecimal notation.

Format

[SET]	**Verify** **ON** [[**Hex**] *startcol* *endcol*] ... **OFF** [[**Hex**] *startcol* *endcol*] ... [**Hex**] *startcol* *endcol* ...

Operands

ON
specifies that all lines changed by subcommands are to be displayed. (This is the initial setting for a typewriter terminal.)

OFF
specifies that lines changed by subcommands are not to be displayed. (This is the initial setting for a display terminal.)

Hex
displays the data in hexadecimal notation.

startcol endcol
is a pair of column numbers that defines an area to be displayed.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults" on page 403.

Usage Notes

1. Up to 28 pairs of columns may be specified. For example:
```
v 1 20 40 50
```
displays columns 1 through 20, and 40 through 50.

2. An area can be displayed in both EBCDIC and hexadecimal. For example:
```
v 1 20 H 1 20
```
displays columns 1 through 20 in both EBCDIC and hexadecimal.

3. The data can be changed in either the EBCDIC or the hexadecimal display. If changed in the hexadecimal display, changes must be entered in hexadecimal. If you type an invalid hexadecimal code, the following message is displayed on the first line of the screen (the file identification line):
```
Invalid hex-data on screen:
```

4. The SET IMAGE setting affects the way that hexadecimal data is treated. An example of this would be, if IMAGE is ON and you enter a X'05', it will expand into X'40's (or to whatever the FILLER character is in hexadecimal) to the next tab position.
5. In multiple views of the same column, if changes are made on the screen the right-most change is the effective one.
6. The columns specified in a SET VERIFY subcommand override any RIGHT or LEFT subcommand that was in effect.
7. The editor displays a file line on as many screen lines as necessary. You can turn off this "automatic line wrapping" feature by issuing the appropriate SET VERIFY subcommand.

   For example, the default VERIFY setting for a SCRIPT file type is 1-132. To display only columns 1-72 of each line, thereby preventing lines longer than 72 characters from wrapping around to the next screen line, you could issue SET VERIFY 1 72. (You could then use a RIGHT subcommand to view the columns of data extending past column 72.)

   Automatic line wrapping is not available on vertical screens.
8. For typewriter terminals, the maximum verify width is 132.

Messages

009E	Column *nn* exceeds record length (*nn*) [RC=5]
520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]
575E	Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} column(s) defined [RC=5]
576E	{Total verify width exceeds screen size (*nn*)|Total offset exceeds LRECL (*nn*)} [RC=5]
581E	Subcommand is not valid in extended mode [RC=3]

Return Codes

0	Normal
3	Subcommand is not valid in extended mode
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET WRAP

Use the WRAP option to control whether or not the editor is to "wrap around" the file if the end of file (or range) is reached during a LOCATE, CLOCATE, FIND, FINDUP, NFIND, or NFINDUP subcommand.

Format

[SET]	**WRap**	**ON** **OFF**

Operands

ON
specifies that the editor is to continue the search up to the line preceding the current line (or following the current line, depending on the search direction). When a CLOCATE is issued, the search continues up to the character preceding the column pointer (or following the column pointer).

OFF
specifies that the editor is to stop searching at the end (or top) of file (or range).

Initial Setting

```
WRAP  OFF
```

Messages

520E	Invalid operand: *operand* [RC=5]
545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET ZONE

Use the ZONE option to define the columns (starting position and ending position) of each record to be scanned for target searches. The editor searches for targets only within the zone.

Format

[SET]	**Zone** *zone* **1** *zone* **2** *****

Operands

zone1
is the starting (left) column number of the zone.

zone2
is the ending (right) column number of the zone. If you specify an asterisk (*), zone2 is the truncation column. (Zone2 cannot be larger than the truncation column.)

Initial Setting

The ZONE is set from the first tab column up to the truncation column.

Usage Notes

1. Column pointer movement is limited by the zone definition: the CFIRST subcommand places the column pointer in column zone1; the CLAST subcommand places the column pointer in column zone2.
2. When a character string that is the target of a search spans several lines and the following subcommand has been issued:

   ```
   set span on noblank
   ```

   Column zone2 of the current line is immediately followed by column zone1 of the next line, with no intervening blank.

   If the following subcommand has been issued:

   ```
   set span on blank
   ```

   Column zone2 of the current line is considered to be separated by a blank from column zone1 of the next line.
3. For a CHANGE subcommand, the string to be changed (string1) is searched for only within the defined zones.

Response

When a SET ZONE subcommand is issued, its effect on column pointer movement is one of the following:

- If the column pointer already lies between the two new zones, it remains unchanged.
- If the current setting of the column pointer is less than zone1, it moves to the column defined by zone1 − 1 (TOL).
- If the current setting of the column pointer is greater than zone2, it moves to the column defined by zone2 + 1 (EOL).

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
575E	Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} columns defined [RC=5]

Return Codes

0	Normal
5	Missing or invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET =

Use the = option to insert the specified string into the equal (=) buffer. (See the = subcommand.)

Format

SET	= *string*

Operand

string
is any XEDIT subcommand or macro, except for the = subcommand (or any CP or CMS command, if SET IMPCMSCP is in effect).

Usage Note

- The subcommand SET is required with this option (to avoid conflict with the = subcommand.

Message

545E	Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SHIFT

Use the SHIFT subcommand to move data to the right or to the left. Data loss is possible.

Format

SHift	**Left** / **Right** [*cols* / 1 [*target* / 1]]

Operands

Left
shifts data to the left. Data shifted to the left past the zone1 column is lost. The line is padded with blanks to the right, up through the truncation column.

Right
shifts data to the right. Shifted data that extends past the truncation column may be lost (See usage note 2, below). The line is padded to the left with blanks.

cols
is the number of columns the data is to be shifted.

target
defines the number of lines to be shifted, starting with the current line, up to but not including the target line. If you enter an asterisk (*), the rest of the file is shifted. If you omit target, only the current line is shifted.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Notes

1. The SHIFT subcommand should not be confused with LEFT, RIGHT, and SET VERIFY, which move the screen over the data, causing the data to *appear* to move in the opposite direction, and do not cause data loss.
2. If SET SPILL OFF is in effect (the default), characters that have been shifted beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been shifted beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. SET SPILL affects only SHIFT issued with the RIGHT operand. SET SPILL has no effect on SHIFT LEFT; data shifted to the left past zone1 is lost.

Responses

If you specify that a SHIFT is to occur on multiple lines and it does occur, the current line pointer will be

a. Unchanged, if SET STAY ON has been issued
b. Moved to the last line shifted, if SET STAY OFF is in effect (the default).

The column pointer is not affected by SHIFT.

Messages

504E	*nn* line(s) {truncated|spilled} [RC=3]
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]
585E	No line(s) changed [RC=1 or 4]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
3	Truncated or spilled
4	No lines changed
5	Invalid or missing operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SI (Macro)

Use the SI (Structured Input) macro to continuously add lines to a file. The first line is added following the line that contains the cursor. The cursor is then positioned at the column where the text on the preceding line begins. Each time that you type on the new line and then press the ENTER key, another new line is automatically added to the file. If nothing is typed on the added line, the line is deleted when the ENTER key is pressed.

Format

SI	

Usage Notes

1. SI can also be issued in the prefix area. See the SI prefix macro.
2. Using SI you can add a blank line to a file by making at least one change on the line. The line is considered changed if you press the space bar or if you retype the mask characters (see SET MASK). Moving the cursor using the cursor position keys over a line does not change the line.
3. After SI is terminated, the cursor is positioned at the indentation column of the last added line.
4. As lines are added using SI, any data above the new line remains stationary while the data below scrolls down the screen. When the new line is one line above the bottom of the file area, adding more lines will scroll the data above the new line up the screen. Using multiple SI subcommands or other prefix subcommands along with SI may move the new line off the screen. The cursor may not be placed at the indentation column when multiple SI subcommands are issued.

Note for Macro Writers

SI uses the console stack to adjust the current line when anything is pending on the line that will become the current line.

Responses

The prefix area of the new line contains '. . . . .' and the following message is displayed in the status area:

```
'.....' pending...
```

This pending status allows you to continually add lines after you have typed on the new line.

By default, the prefix area for the line that is added is highlighted. For more information, refer to SET COLOR PENDING.

Each line that is added is pre-filled with the current mask (see SET MASK).

Messages

520E	Invalid operand: *operand* [RC=5]
529E	SI is only valid in {display|editing} mode [RC=3]
561E	Cursor is not on a valid data field [RC=1 or 3]

Return Codes

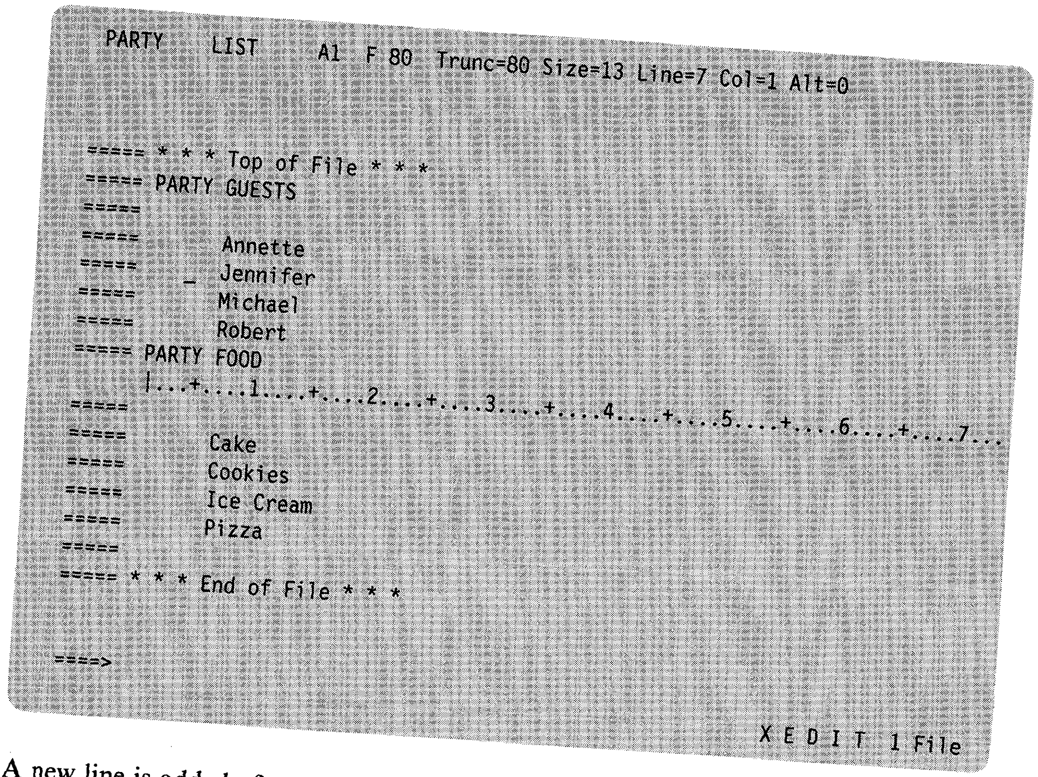
3	Cursor is not in a valid field, or SI is valid in display mode only
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

The following example shows how to use a PF key assigned the SI subcommand.

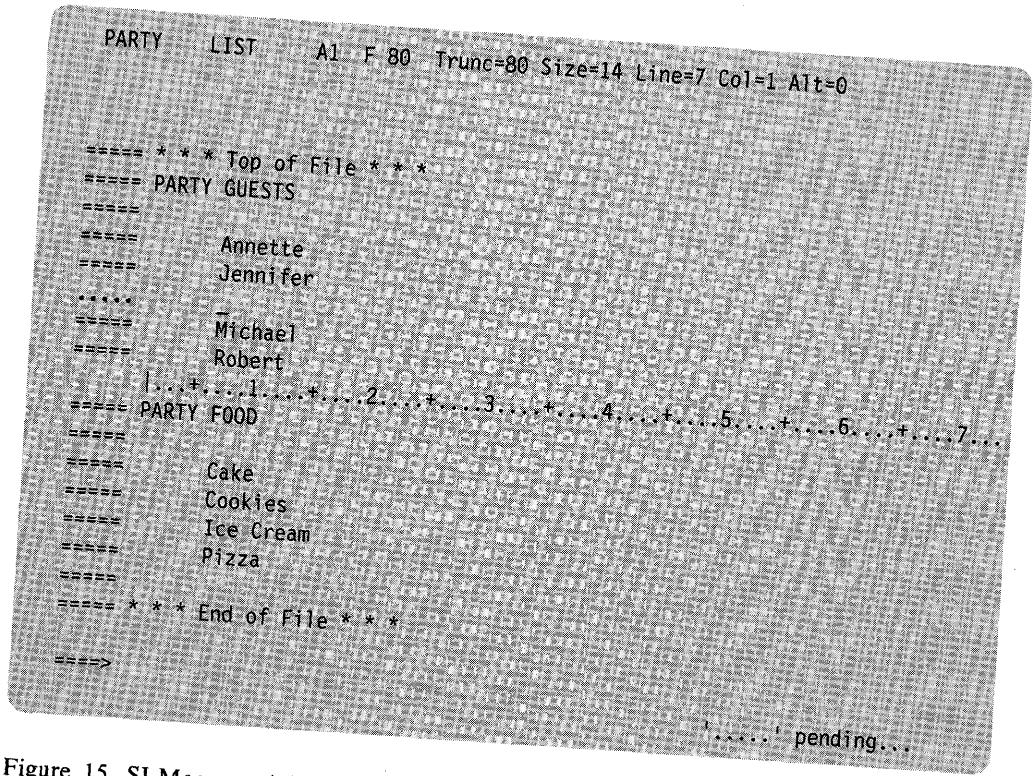
To insert the names of more party guests into the following file:

- Move the cursor to the line where you want to add a name.
- Press the PF key assigned to SI.



```
  PARTY    LIST     A1  F 80  Trunc=80 Size=13 Line=7 Col=1 Alt=0


===== * * * Top of File * * *
===== PARTY GUESTS
=====
=====       Annette
=====     _ Jennifer
=====       Michael
=====       Robert
===== PARTY FOOD
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====       Cake
=====       Cookies
=====       Ice Cream
=====       Pizza
=====
===== * * * End of File * * *

====>
                                                              X E D I T  1 File
```

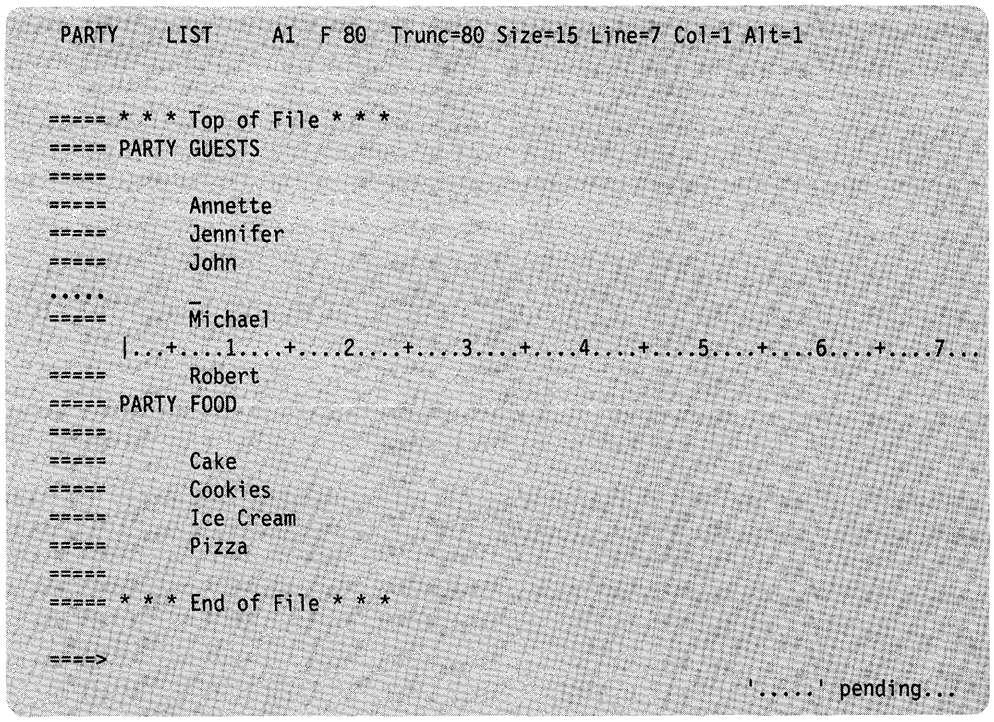
A new line is added after the line the cursor is on. The prefix area of the new line contains ". . . . ." and the cursor is positioned at the indentation column of the previous line.



```
  PARTY    LIST     A1  F 80  Trunc=80 Size=14 Line=7 Col=1 Alt=0


===== * * * Top of File * * *
===== PARTY GUESTS
=====
=====       Annette
=====       Jennifer
.....
            _
=====       Michael
=====       Robert
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== PARTY FOOD
=====
=====       Cake
=====       Cookies
=====       Ice Cream
=====       Pizza
=====
===== * * * End of File * * *

====>
                                                          '.....' pending...
```

Figure 15. SI Macro – Adding the First New Line

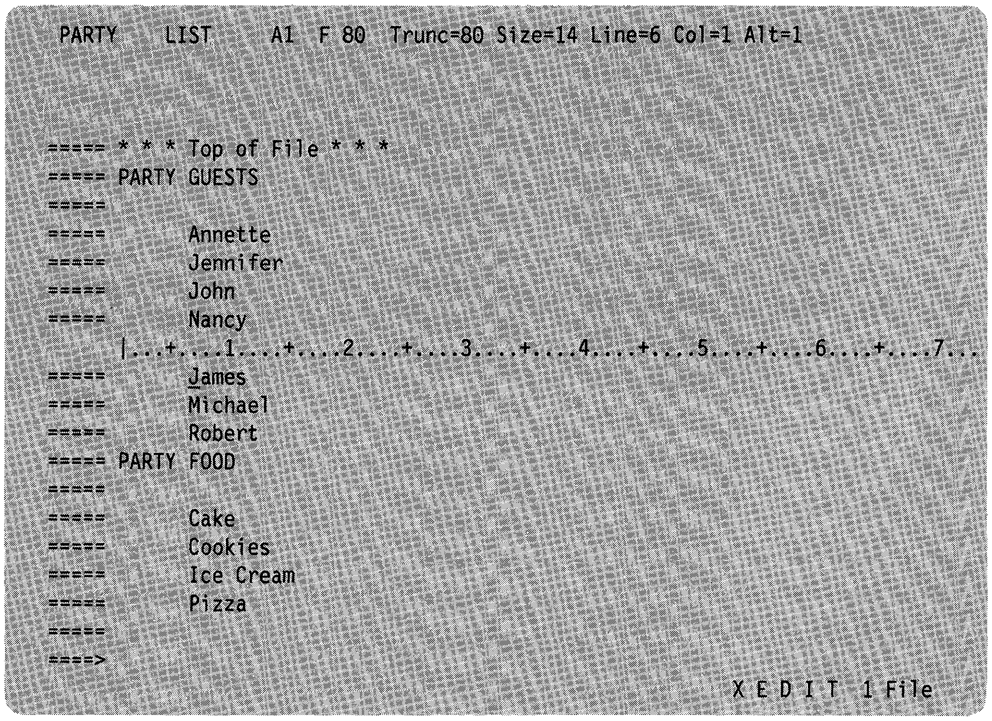
Type a name on the line and press the ENTER key. This results in a new line being added following the one you just typed on. You can continue to enter names to the list by typing on the new line and pressing the ENTER key.



```
 PARTY    LIST     A1  F 80  Trunc=80 Size=15 Line=7 Col=1 Alt=1


===== * * * Top of File * * *
===== PARTY GUESTS
=====
=====       Annette
=====       Jennifer
=====       John
.....       _
=====       Michael
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====       Robert
===== PARTY FOOD
=====
=====       Cake
=====       Cookies
=====       Ice Cream
=====       Pizza
=====
===== * * * End of File * * *

====>
                                                               '.....' pending...
```

Figure 16 (Part 1 of 2). SI Macro — Adds New Lines Until Null Line Entered

When you are finished adding names, then press the ENTER key without typing on the new line. In this example two more names were added before terminating SI.



```
  PARTY    LIST     A1  F 80  Trunc=80 Size=14 Line=6 Col=1 Alt=1



 ===== * * * Top of File * * *
 ===== PARTY GUESTS
 =====
 =====       Annette
 =====       Jennifer
 =====       John
 =====       Nancy
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
 =====       James
 =====       Michael
 =====       Robert
 ===== PARTY FOOD
 =====
 =====       Cake
 =====       Cookies
 =====       Ice Cream
 =====       Pizza
 =====
 ====>
                                                          X E D I T  1 File
```

Figure 16 (Part 2 of 2). SI Macro — Adds New Lines Until Null Line Entered

SORT (Macro)

Use the SORT macro to arrange a specified number of file lines in ascending or descending EBCDIC order according to specified sort columns.

Format

SORT	*target* [<u>A</u>] [D] *col* 1 *col* 2 [*col* 1 *col* 2] ...

Operands

target
specifies the number of lines to be sorted. Lines are sorted starting with the current line, up to but not including the target line. (If you specify an asterisk (*), lines are sorted starting with the current line to the end of the file.)

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

<u>**A**</u>
sorts the file in ascending EBCDIC order. This is the default.

D
sorts the file in descending EBCDIC order.

col1 col2
is a pair of numbers that define a sort field. Col1 is the starting column of a sort field within each line. Col2 is the ending column. You can specify as many sort fields as you want, as long as the total length of the fields does not exceed 249.

Usage Notes

1. If SET CASE UPPERCASE/MIXED RESPECT has been issued, sorting is done in EBCDIC order.
2. If SET CASE UPPERCASE/MIXED IGNORE has been issued, sorting is done in alphabetical order, with lowercase and uppercase representations of the same letter considered to be identical.
3. SORT operates outside of the current SCOPE. (See Appendix B for more information about SORT and the SCOPE setting.)
4. SORT cannot be issued when prefix subcommands or macros are pending and it cannot be issued from a prefix macro.

Messages

009E	Column exceeds record length [RC=24]
053E	Invalid sort field pair defined [RC=24]
063E	No sort list given [RC=40]
493E	SORT invalid in update mode [RC=3]
545E	Missing operand(s) [RC=5]
546E	Target not found [RC=2]

581E	Subcommand is not valid in extended mode [RC=3]
588E	Prefix subcommand waiting... [RC=8]
596E	This module must be called within the editor [RC=88]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
3	SORT cannot be used when a file is edited in UPDATE mode or extended mode
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Prefix area contains pending subcommand or macro
24	Invalid columns defined
40	No list given
88	Module must be called within the editor

Example

Suppose you are editing a file which has two fields of information. The first field, positioned between columns 8 and 16 of the file, contains a list of file names. The second field, positioned between columns 17 and 24 of the file, contains a list of file types. Issuing the command:

```
sort * 17 24 8 16
```

sorts the file by file type (columns 17 through 24) and, within each file type, by file name (columns 8 through 16). The asterisk tells XEDIT to sort the entire file. Because no sort order is specified, the file will be sorted in ascending order (the default).

SOS

The SOS (screen operation simulation) subcommand provides a set of functions to be used mainly in XEDIT macros or to be assigned to PF keys.

Format

SOS	*option*	
	<u>**Options:**</u>	
	Alarm	**POP**
	CLEAR	**PUsh**
	LINEAdd	**TABB** [*n* | 1]
	LINEDel	**TABCmd**
	NUlls	**TABCMDB** [*n* | 1]
	NUlls ON	**TABCMDF** [*n* | 1]
	NUlls OFF	**TABF** [*n* | 1]
	PFn	

Operands

ALarm
Ring the terminal alarm the next time the display is refreshed.

CLEAR
Clears the screen.

LINEAdd
Add a blank line after the line pointed to by the cursor. This is the initial setting of the PF2 key.

LINEDel
Delete the line pointed to by the cursor.

NUlls
Reverse the current setting of the "nulls" option for the line pointed to by the cursor.

NUlls ON
Change the trailing blanks to "nulls" for the line pointed to by the cursor.

NUlls OFF
Change the trailing "nulls" to blanks for the line pointed to by the cursor.

PF*n*
Depress a PF key where n is the key number 1-24. The data which had been assigned to the key is placed LIFO in the CMS console stack.

POP
Remove the top position from the cursor stack and place the cursor there. (See PUSH, below.) If the cursor is outside any logical screen, it is positioned in the upper left corner of the first logical screen.

PUsh
Save the current position of the cursor. The position is saved in a LIFO fashion in a five-position stack used only for this purpose.

TABB [*n*|<u>1</u>]
: Move the cursor backward to the previous "tab" position as indicated by the XEDIT subcommand, SET TABS. The "tab" operation may be performed n times with one (1) being the default.

TABCmd
: Position the cursor at the command line for the logical screen in which it currently resides.

TABCMDB [*n*|<u>1</u>]
: Move the cursor backward to the first encountered command line. When multiple logical screens are defined, moves the cursor to the command line of the previous logical screen.

TABCMDF [*n*|<u>1</u>]
: Move the cursor forward to the first encountered command line. When multiple logical screens are defined, moves the cursor to the command line of the next logical screen.

TABF [*n*|<u>1</u>]
: Move the cursor forward to the next "tab" position as indicated by the XEDIT subcommand, SET TABS. The "tab" operation may be performed n times with one (1) being the default.

Usage Notes

1. When the prefix is set on the left and the cursor is placed in the attribute byte following the line, TABF moves the cursor to the first tab column in the same line and TABB moves the cursor to the last tab column of the preceding line.
2. When using TABCMDF or TABCMDB, the view containing the first encountered command line becomes the file currently being edited. This switches editing to the next view displayed. As a result, if a TABCMDF or TABCMDB is issued when the commands are stacked (i.e., by a PF key), the stacked commands will be executed on the view containing the first encountered command line.
3. With ETMODE ON, TABF with the cursor on the command line can produce unpredictable results.

Messages

520E	Invalid operand: *operand* [RC=5]
529E	Subcommand is only valid in {display|editing} mode [RC=3]
545E	Missing operand(s) [RC=5]
561E	Cursor is not on a valid data field [RC=1 or 3]

Return Codes

0	Normal
1	Cursor is on TOF or EOF line
3	Invalid placement of cursor or subcommand, or subcommand is valid only for display terminal
5	Missing or invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SPLIT (Macro)

Use the SPLIT macro to split a line into two or more lines.

The first format allows you to split a line into two lines, at the column pointer or at the cursor.

The second format allows you to split a line into several lines.

Format

SPlit	**[ALigned]** [**Column** / **CURSOR**]
	[ALigned] [*colno* / [**Before** / **After**] */string/*] ...

Operands

ALigned
gives the created line the same number of leading blanks as the original line.

Column
splits the current line into two lines. The second line starts with the data in the current column (as defined by the column pointer). The column pointer remains unchanged. If SPLIT is entered without an operand, SPLIT COLUMN is the default.

CURSOR
splits the line containing the cursor into two lines. The second line starts with the character under which the cursor was positioned. The cursor is not moved. This format of the SPLIT macro is especially useful when assigned to a PF key.

colno
specifies that the current line is to be split at the specified column number(s). The line is split as many times as there are operands.

Before
splits the current line *before* a specified character string. This is the default.

After
splits the current line *after* a specified character string.

/string/
splits the current line before or after the specified character string. The line is split as many times as there are operands.

Usage Notes

1. The SPLIT macro searches for strings in the current line with VARBLANK, SPAN, and STREAM all set to OFF (and restored after the SPLIT).
2. SPLIT is the converse of JOIN. (See also SPLTJOIN, which combines the functions of SPLIT and JOIN.)
3. The cursor or the column number or string specified must fall within the current zones.

Responses

The second line begins in column 1 unless SET IMAGE ON is in effect, in which case the second line begins in the first tab column.

If the current line is split, the last line added becomes the new current line. If SPLIT CURSOR is issued, the line pointer remains unchanged.

If a specified string is not found, an error message is issued and the current line remains the same.

Messages

526E	Option *option* valid in display mode only [RC=3]
557S	No more storage to insert lines [RC=4]
561E	Cursor is not on a valid data field [RC=1 or 3]
575E	Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} column(s) defined [RC=5]
585E	No line(s) changed [RC=1 or 4]
586E	Not found [RC=2]

Return Codes

0	Normal
1	No change (SPLIT issued at TOF or EOF)
2	Not found
3	Invalid placement of cursor or subcommand, or subcommand is valid only for display terminal
4	No more storage
5	Invalid operands
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

Current Line:

```
===== Electric eels can discharge bursts of 625 volts.
```

Note the cursor position above. Press a PF key assigned to SPLIT CURSOR.

```
===== Electric eels can discharge _
===== bursts of 625 volts.
```

Current Line:

```
===== Electric eels can discharge bursts of 625 volts.
```

`sp 29` (First line contains columns 1 through 28.)

```
===== Electric eels can discharge
===== bursts of 625 volts.
```

Current Line:

```
=====     Electric eels can discharge bursts of 625 volts.
```

`sp aligned 23 (Split at column 23, but line up with first line.)`

```
=====     Electric eels can
=====     discharge bursts of 625 volts.
```

Current Line:

```
===== Electric eels can discharge bursts of 625 volts.
```

`sp a/discharge /` (Split the line after "discharge .")

```
===== Electric eels can discharge
===== bursts of 625 volts.
```

Current Line:

```
===== Electric eels can discharge bursts of 625 volts.
```

`sp 15 b/bursts/` (Split the line into three lines.)

```
===== Electric eels
===== can discharge
===== bursts of 625 volts.
```

The new current line is the last one that was added as a result of the split.

SPLTJOIN (Macro)

Use the SPLTJOIN macro either to split a line or join two lines, depending on the position of the cursor on a file line. If the cursor is positioned before or at the last non-blank character, the line is split (at the cursor position). If the cursor is positioned after the last non-blank character on a line (that is, after the end of the data on a line), the next line is appended, starting at the cursor position.

Format

SPLTJOIN	

Usage Notes

1. The SPLTJOIN macro is most useful when assigned to a PF key. The PF11 key is initially set to SPLTJOIN.
2. SPLTJOIN enables display terminal users to combine the functions of SPLIT CURSOR and JOIN CURSOR (see SPLIT and JOIN) on a single PF key. Unlike JOIN CURSOR, it prevents the accidental loss of data caused by joining two lines and overlaying data.
3. SPLTJOIN functions like SPLIT and JOIN issued with the ALIGNED operand, that is, it "takes care of" leading blanks. If a line is split, it adds the same number of leading blanks to the beginning of the new line as the original line has. If two lines are joined, it removes the same number of leading blanks from the line being joined as there are on the line to which it is appended.
4. The cursor must lie with the current zones.
5. If SET SPILL ON or SET SPILL WORD has been issued, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. For the SPLTJOIN macro, SET SPILL OFF (the default) has the same effect as SET SPILL ON. SPLTJOIN does not truncate data.
6. If two lines are joined, the original line appended as a result of the join is deleted. (The line pointer remains unchanged.)

Messages

503E	{Truncated|Spilled} [RC=3]
520E	Invalid operand: *operand* [RC=5]
529E	SPLTJOIN is only valid in {display|editing} mode [RC=3]
561E	Cursor is not on a valid data field [RC=1 or 3]
585E	No line(s) changed [RC=1 or 4]
557S	No more storage to insert lines [RC=4]

Return Codes

0	Normal
1	No change (SPLTJOIN issued at TOF or EOF)
3	Spill occurred, or invalid placement of cursor or subcommand, or subcommand is valid only for display terminal
4	No more storage
5	Invalid operand
6	Subcommand rejected in the PROFILE due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

STACK

Use the STACK subcommand to place part or all of a specified number of lines (FIFO) in the console stack, starting with the current line. This subcommand is designed to be issued from a macro.

Format

```
STAck    [ target   [ startcol   [ length ]]]
         [   1      [    1       [   *    ]]]
```

Operands

target
: defines the number of lines to be stacked. Lines are stacked starting with the current line, up to but not including the target line. If you specify an asterisk (*), the rest of the file is stacked. If you omit target, only the current line is stacked.

  A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the *VM/SP System Product Editor User's Guide.*.

startcol
: specifies that lines are to be stacked starting in this column number. If you omit startcol, the line is stacked starting at the first column.

length
: specifies the number of columns that are to be stacked, starting with startcol. The maximum value for length is the truncation column. If you omit the length, the line is stacked up to the truncation column.

Notes for Macro Writers

1. STACK 1 1 0 stacks an empty line. STACK 0 also stacks an empty line.
2. The CMS console stack restricts the length to be stacked to 255 bytes. To retrieve information about lines that are longer than 255 bytes, use EXTRACT/CURLINE/ in a System Product Interpreter macro.
3. The line pointer is moved to the last line stacked.
4. If a backward target is specified, for example, −3, the lines are stacked in reverse order.

Messages

520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]
546E	Target not found [RC=2]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

STATUS (Macro)

Use the STATUS macro to display the SET subcommand options and their current settings, or to create an XEDIT macro that contains the SET subcommand options listed under Usage Note 2.

Format

STATus	[*filename*]

Operand

filename
specifies the name of a file that is to contain all the SET subcommand options listed in usage note 2.

The editor assigns a file type of XEDIT; therefore, the file is an XEDIT macro. The macro can be invoked later to restore the SET subcommand options that were in effect when the STATUS macro was issued.

Usage Notes

1. Use the STATUS macro without an operand to display the SET subcommand options and their current settings. All SET subcommand options are displayed except:

```
ALT       PF
BRKKEY    POINT
ENTER     RESERVED
ETARBCH   SELECT
LASTLORC  SIDCODE
PA        TRANSLAT
PENDING   =
```

2. When you use the STATUS macro to create an XEDIT macro, you can later invoke the macro to restore the SET subcommand options that were in effect when the STATUS macro was issued.

   For example:

```
status KEEP
```

   KEEP is the name of the macro.

Settings of the following subcommands are included in the macro:

APL	FNAME	NULLS	SPILL
ARBCHAR	FTYPE	NUMBER	STAY
AUTOSAVE	FULLREAD	PACK	STREAM
CASE	HEX	PREFIX	SYNONYM
CMDLINE	IMAGE	RANGE	TABLINE
COLOR	IMPCMSCP	RECFM	TABS
COLPTR	LINEND	REMOTE	TERMINAL
CTLCHAR	LRECL	SCALE	TEXT
CURLINE	MACRO	SCOPE	TOFEOF
DISPLAY	MASK	SCREEN	TRUNC
ESCAPE	MSGLINE	SERIAL	VARBLANK
ETMODE	MSGMODE	SHADOW	VERIFY
FILLER	NONDISP	SPAN	WRAP
FMODE			ZONE.

Response

```
703I File filename XEDIT A1 created
```

Messages

024E	File *fn* XEDIT A already exists [RC=28]
520E	Invalid operand: *operand* [RC=5]
671E	Error {sending|receiving|creating|loading|updating} [file] *fn ft fm*, rc=*nn* from [RC=100]

Return Codes

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
28	File name specified already exists
100	Error occurred while creating the file

TOP

Use the TOP subcommand to move the line pointer to the null line above the first line of the file or of the range (see the SET RANGE subcommand).

Format

TOP	

Usage Note

TOP is the proper subcommand to use before a subcommand that searches for the first occurrence of a string in a file (such as LOCATE, FIND, etc.). If the current line is the first line of the file, a string occurring in this line would be missed, because LOCATE, FIND, etc. start searching with the line *following* the current line.

Responses

The null line at the top of the file becomes the new current line and contains:

```
* * * Top of File * * *
```

The lines preceding the current line are blank, and the rest of the screen contains the beginning lines of the file.

Message

520E Invalid operand: *operand* [RC=5]

Return Codes

1	Top of file reached
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

TRANSFER

The TRANSFER subcommand is used *within a macro* to access specified editing variables, for example, the current line number, the file name of the file being edited, etc. The values that are transferred are placed in the console stack and can be read by subsequent EXEC 2 &READ control statements. More than one keyword can be specified in one TRANSFER subcommand placing the values last-in-first-out (LIFO) on the console stack.

Only VM/SP Release 1 and Release 2 settings can be transferred. For example, *TRANSFER CTLCHAR char* does not transfer *color* and *exthi* attributes introduced in Release 3. The EXTRACT subcommand returns *all* settings and provides greater flexibility.

Format

```
TRAnsfer    APL                  PFn
            ARBchar              Point
            AUtosave             PREfix
            CASE                 RANge
            CMDline              RECFm
            COLPtr               RESERved
            COLumn               SCALe
            CTLchar [char]       SCReen
            CURLine              Seq8
            CURSor               SERial
            EOF                  SIDcode
            ESCape               SIZe
            FILler               SPAN
            FMode                STAY
            FName                STReam
            FType                SYNonym [name]
            HEX                  TABLine
            IMage                TABS
            IMPcmscp             TARGet
            LASTmsg              TERMinal
            LENgth               TEXT
            LIne                 TOF
            LINENd               TOFEOF
            LRecl                TRunc
            LScreen              UPDate
            MACRO                VARblank
            MASK                 Verify
            MSGMode              VERShift
            NBFile               Width
            NONDisp              WRap
            NULls                Zone
            NUMber               =
            PACK
```

Operands

APL
transfers "ON" or "OFF" as defined by the SET APL subcommand.

ARBchar
transfers "ON" or "OFF" and the arbitrary character specified in the SET ARBCHAR subcommand.

AUtosave
transfers the current AUTOSAVE setting: the AUTOSAVE count, file ID, and number of alterations.

CASE
transfers two values: the case setting ("U" or "M") and "R" or "I" as defined in the SET CASE subcommand.

CMDline
transfers an integer (n), which is the screen command line number defined by the SET CMDLINE subcommand. If SET CMDLINE TOP, n=2. If SET CMDLINE ON, the number is the logical screen size minus one. If SET CMDLINE BOTTOM, n is the number of the last line on the logical screen. If SET CMDLINE OFF, n=0.

COLPtr
transfers "ON" or "OFF" as defined by the SET COLPTR subcommand.

COLumn
transfers the column number of the column pointer.

CTLchar [*char*]
If no character is specified (TRA CTLCHAR), transfers the control character identifier and all characters defined by the SET CTLCHAR subcommand, in the form "CTLCHAR ON ESCAPE char CTL c1 c2 c3 c4. . .". If no control characters are defined, transfers "CTLCHAR OFF."

If a character is specified (TRA CTLCHAR char), the attributes of that character are transferred, in the form "CTLCHAR char attribute1 PROTECT|NOPROTECT attribute2 HIGH|NOHIGH|INVISIBLE." If no attributes were defined for the character, transfers "CTLCHAR."

If the TRANSFER subcommand specifies multiple operands, the operand that follows CTLCHAR is interpreted as follows: if it is one character in length, it is interpreted as the "char" operand of CTLCHAR; if it is longer than one character or is not specified, it is handled normally.

For example:

```
transfer fn ft fm ctlchar lrecl recfm
```

CTLCHAR is treated as if it were specified without the "char" operand. LRECL and RECFM are handled normally.

```
transfer ctlchar ¢ ctlchar " ctlchar % lrecl recfm
```

transfers the attributes of ¢, ", and %.

CURLine
transfers the line number of the current line relative to the top of the screen, as defined by the SET CURLINE subcommand.

CURSor
transfers four integers: the current position of the cursor on the screen (line number and column number) and the position of the cursor in the file (line number and column number). If the cursor is in a protected area, two negative numbers (−1) are transferred for the position of the cursor in the file. The top and bottom of the range are considered to be in the file.

The current position of the cursor is the location where the cursor would be placed if the screen were displayed at this time.

EOF
transfers "ON" or "OFF" as determined by the editor. EOF is "ON" when the line pointer reaches end of file.

ESCape
transfers "ON" or "OFF" and the escape character (one-character string) defined by the SET ESCAPE subcommand. This character may be blank.

FILler
transfers the filler character (one-character string) defined by the SET FILLER subcommand. This character may be blank.

FMode
transfers the two-character file mode.

FName
transfers the eight-character file name.

FType
transfers the eight-character file type.

HEX
transfers "ON" or "OFF" as specified in the SET HEX subcommand.

IMage
transfers "ON," "OFF," or "CANON" as specified in the SET IMAGE subcommand.

IMPcmscp
transfers "ON" or "OFF" as specified in the SET IMPCMSCP subcommand.

LASTmsg
transfers the last message issued by the editor. This message may or may not have been displayed, depending on the SET MSGMODE subcommand operands.

LENgth
transfers the length of the current line from column one through the truncation column, excluding trailing blanks.

LIne
transfers the current line number, relative to the beginning of the file.

LINENd
transfers "ON" or "OFF" and the line end character as defined by the SET LINEND subcommand.

LRecl
transfers the logical record length.

LScreen
transfers six integers: the number of lines and the number of columns of the logical screen; the line number and column number defining the top left corner of the logical screen on the virtual screen; the number of lines and number of columns of the virtual screen.

MACRO
transfers "ON" or "OFF" as specified by the SET MACRO subcommand.

MASK
transfers the current mask line as defined by the SET MASK subcommand. The line may be all blanks.

MSGMode
transfers "ON" or "OFF" and "LONG" or "SHORT" as defined by the SET MSGMODE subcommand.

NBFile
transfers the number of files being edited.

NONDisp
transfers the character defined by the SET NONDISP subcommand. The character may be blank.

NULls
transfers "ON" or "OFF" as specified by the SET NULLS subcommand.

NUMber
transfers "ON" or "OFF" as specified by the SET NUMBER subcommand.

PACK
transfers "ON" or "OFF" as specified by the SET PACK subcommand.

PFn
transfers the string associated with a specified PF key, as defined by SET PFn. The string may be null or blank.

Point
transfers the symbolic name associated with the current line, as defined by the SET POINT subcommand or the .xxxx prefix subcommand, or transfers a blank string if no name has been defined.

PREfix
transfers "ON," "OFF," or "NULLS" and "RIGHT" or "LEFT" as specified in the SET PREFIX subcommand.

RANge
transfers two integers, which are the line numbers of the top and bottom of the range defined by the SET RANGE subcommand.

RECFm
transfers the record format, "F," "V," "FP," or "VP," defined by the SET RECFM subcommand.

RESERved
transfers as one line, the line numbers of reserved lines.

SCALe
transfers "ON" or "OFF" and the scale line number as specified in the SET SCALE subcommand.

SCReen
transfers "SIZE n1 n2. . .," where n1 is the number of lines in the first logical screen, n2 is the number of lines in the second logical screen, etc., as defined by the SET SCREEN subcommand.

Seq8
transfers "OFF" if the XEDIT command was issued with the NOSEQ8 operand; if not, transfers "ON."

SERial
transfers the serial identification or "OFF," the increment value, and the serial number starting value as defined by the SET SERIAL subcommand.

SIDcode
transfers the eight-character string specified in the SIDCODE option of the XEDIT command (or subcommand) or the LOAD subcommand.

SIZe
transfers the number of records in the file being edited.

SPAN
transfers three values: "ON" or "OFF," "B" or "N," and n, as defined in the SET SPAN subcommand.

STAY
transfers "ON" or "OFF" as specified in the SET STAY subcommand.

STReam
transfers "ON" or "OFF" as specified in the SET STREAM subcommand.

SYNonym [*name*]
TRANSFER SYNONYM transfers "ON" or "OFF" as specified in the SET SYNONYM subcommand. TRANSFER SYNONYM *name* transfers the name, its minimum abbreviation, and the associated synonym definition (that is, everything that was entered in the SET SYNONYM subcommand after the minimum abbreviation size).

Note: In a TRANSFER subcommand with multiple keyword operands, SYNONYM must be the last one; otherwise, the keyword following SYNONYM would be interpreted as its operand and not as an independent keyword.

TABLine
transfers "ON" or "OFF" and n, as defined in the SET TABLINE subcommand.

TABS
transfers the tab column numbers defined by the SET TABS subcommand.

TARGet
transfers two pairs of integers pertaining to the character string that matches the last target located: line and column number of the first character in the string; line and column number of the last character in the string.

TERMinal
transfers "DISPLAY" or "TYPEWRITER" as defined in the SET TERMINAL subcommand.

TEXT
transfers "ON" or "OFF" as specified in the SET TEXT subcommand.

TOF
: transfers "ON" or "OFF" as determined by the editor. TOF is "ON" when the current line pointer reaches the top of file.

TOFEOF
: transfers "ON" or "OFF" as specified in the SET TOFEOF subcommand.

TRunc
: transfers the truncation column number as defined by the SET TRUNC subcommand.

UPDate
: transfers "ON" or "OFF" as determined by the editor. Update is "ON" when the XEDIT command has been issued with the UPDATE or CTL operands.

VARblank
: transfers "ON" or "OFF" as specified in the SET VARBLANK subcommand.

Verify
: transfers "ON" or "OFF" and the verification columns specified in the SET VERIFY subcommand.

VERShift
: transfers +n or −n, which is the relative position of the screen over the file, as a result of any LEFT or RIGHT subcommands.

Width
: transfers the WIDTH value as specified in the XEDIT command or LOAD subcommand or as determined by the editor.

WRap
: transfers "ON" or "OFF" as specified in the SET WRAP subcommand.

Zone
: transfers the left and right zone column numbers specified in the SET ZONE subcommand.

=
: transfers one line (up to 130 characters) that corresponds to the content of the "equal (=) buffer." The equal buffer contains the last executed subcommand or macro or CP/CMS command, or whatever has been specified in the SET = subcommand.

Notes for Macro Writers

1. Several values can be transferred in one TRANSFER subcommand. For example:

   ```
   transfer line size trunc
   ```

   The values can then be read by the macro. For example:

   ```
   &read vars &line &size &trunc
   ```

   The variables in the macro will now contain the current line number, the size of the file, and the truncation column, respectively.

2. Remember that some TRANSFER keywords are associated with a *set* of values (like CURSOR and TABS), or a character that may be blank (like ESCAPE), or a text line of varying length (like LASTMSG). When using TRANSFER with one of these keywords, it may be preferable *not* to specify multiple keywords in one TRANSFER and to make proper use of the &READ VARS, &READ ARGS, or &READ STRING control statements.

Message

545E Missing operand(s) [RC=5]

Return Codes

0	Normal
5	Missing operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

TYPE

Use the TYPE subcommand to display a specified number of lines, starting with the current line.

Format

Type	[*target* |1|]

Operand

target
defines the number of lines to be displayed. Lines are displayed starting with the current line, up to but not including the target line. If you enter an asterisk (*), the rest of the file is displayed. If you omit target, only the current line is displayed.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Notes

1. TYPE displays a line according to the current SET VERIFY subcommand.
2. TYPE displays the first 256 characters for each line in the message area. If the output exceeds the number of lines for the message line, then the output is passed to CMS. When full-screen CMS is ON the output will appear in the CMSOUT window. To see all of the information in the virtual screen, you can use the CMS SCROLL command. The screen will be cleared automatically when you scroll to the bottom of the virtual screen. An alternative way to clear the screen is to issue the CMS DROP WINDOW command. When full-screen CMS is OFF, press the CLEAR key to redisplay the file.
3. If SET SHADOW ON and SET SCOPE DISPLAY are in effect, any shadow lines within the lines being typed are also displayed:

   ```
   ------------- nn line(s) not displayed -------------
   ```

   However, shadow lines are not included in the count of lines to be typed.

Responses

The specified lines are displayed.

The line pointer moves to the last line typed.

If the line pointer has moved to the null End of File line, the last line displayed will be:

```
583I  EOF:
```

If the current line is the null Top of File and you issue the TYPE subcommand, the first line displayed will be:

```
584I  TOF:
```

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]

Return Codes

0	Normal
1	TOF or EOF reached
2	Target line not found
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

UP

Use the UP subcommand to move the line pointer a specified number of lines toward the top of the file.

Format

Up	[*n* | * |**1**]

Operand

n

is the number of lines the line pointer is to be moved toward the top of the file. If you specify an asterisk (*), the line pointer moves to the "Top of File" line. If a number is not specified, the pointer is moved up only one line.

Usage Note

The UP subcommand is equivalent to a minus (−) target. For example:

```
up 3
```

is equivalent to

```
-3
```

Response

The line pointed to becomes the new current line.

Messages

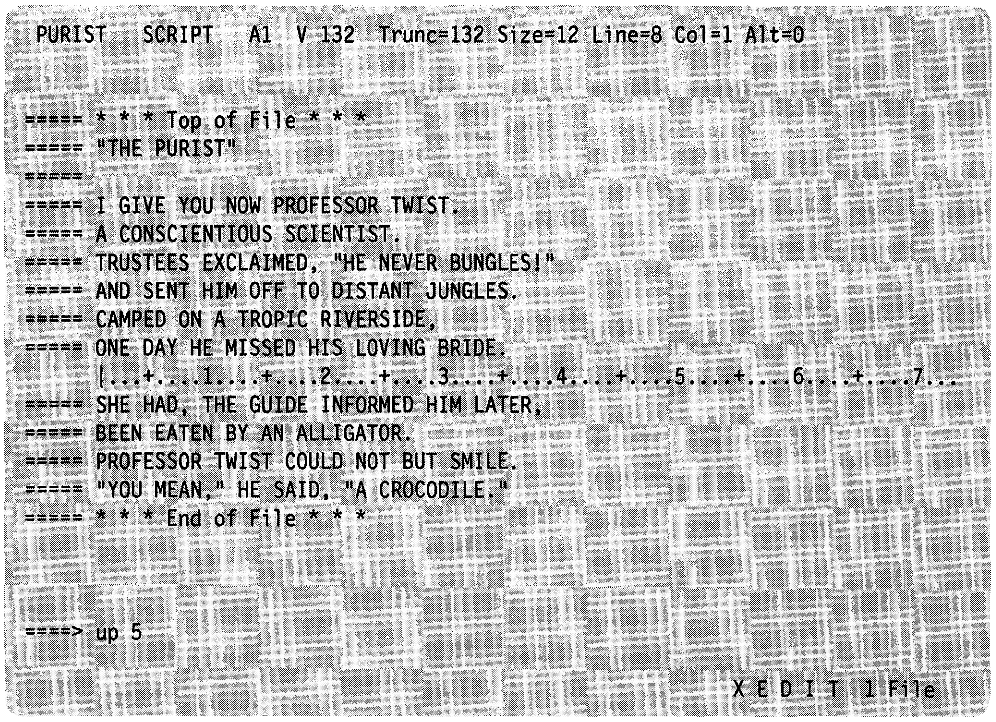
520E	Invalid operand: *operand* [RC=5]
543E	Invalid number: *number* [RC=5]

Return Codes

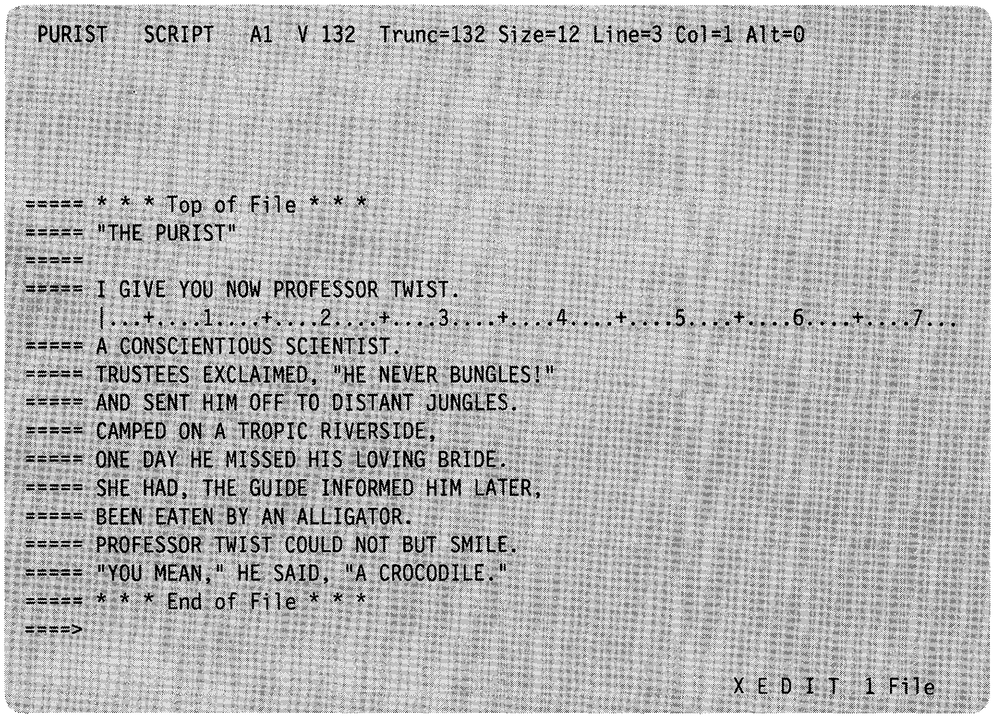
0	Normal
1	Top of File reached and displayed
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

Example

Figure 17 is a before-and-after example of the UP subcommand.



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=8 Col=1 Alt=0


===== * * * Top of File * * *
===== "THE PURIST"
=====
===== I GIVE YOU NOW PROFESSOR TWIST.
===== A CONSCIENTIOUS SCIENTIST.
===== TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
===== AND SENT HIM OFF TO DISTANT JUNGLES.
===== CAMPED ON A TROPIC RIVERSIDE,
===== ONE DAY HE MISSED HIS LOVING BRIDE.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== SHE HAD, THE GUIDE INFORMED HIM LATER,
===== BEEN EATEN BY AN ALLIGATOR.
===== PROFESSOR TWIST COULD NOT BUT SMILE.
===== "YOU MEAN," HE SAID, "A CROCODILE."
===== * * * End of File * * *



====> up 5

                                                               X E D I T  1 File
```



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=3 Col=1 Alt=0





===== * * * Top of File * * *
===== "THE PURIST"
=====
===== I GIVE YOU NOW PROFESSOR TWIST.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== A CONSCIENTIOUS SCIENTIST.
===== TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
===== AND SENT HIM OFF TO DISTANT JUNGLES.
===== CAMPED ON A TROPIC RIVERSIDE,
===== ONE DAY HE MISSED HIS LOVING BRIDE.
===== SHE HAD, THE GUIDE INFORMED HIM LATER,
===== BEEN EATEN BY AN ALLIGATOR.
===== PROFESSOR TWIST COULD NOT BUT SMILE.
===== "YOU MEAN," HE SAID, "A CROCODILE."
===== * * * End of File * * *
====>

                                                               X E D I T  1 File
```

Figure 17. The UP Subcommand — Before and After

UPPERCAS

Use the UPPERCAS subcommand to translate all lowercase characters to uppercase, starting at the current line, for a specified number of lines.

Format

UPPercas	[*target* |1]

Operand

target
defines the number of lines to be translated. Translation starts with the current line and continues up to, but does not include, the target line. If you specify an asterisk (*), the rest of the file is translated. If you omit target, only the current line is translated.

A target may be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, refer to the LOCATE subcommand in this book and to the publication *VM/SP System Product Editor User's Guide.*

Usage Note

The case setting defined in the SET CASE subcommand is not changed by UPPERCAS.

Responses

1. When you press the ENTER key, all lowercase letters within the current zones appear in uppercase.
2. If you specify that UPPERCAS is to occur on multiple lines and it does occur, the current line pointer will be:
   a. Unchanged if SET STAY ON has been issued.
   b. Moved to the last line translated if SET STAY OFF is in effect (the default).

Messages

520E	Invalid operand: *operand* [RC=5]
546E	Target not found [RC=2]
585E	No line(s) changed [RC=4]

Return Codes

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
4	No lines changed
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error,or QUIT subcommand has been issued in a macro called from the last file in the ring

Examples

```
===== Elephant tusks can weigh up to 300 pounds.
```

`upp` (translate the current line to uppercase)

```
===== ELEPHANT TUSKS CAN WEIGH UP TO 300 POUNDS.
```

XEDIT

Use the XEDIT subcommand to edit multiple files.

Format

Xedit	[*fn* [*ft* [*fm*]]] [(*options...* [)]]

Operands

fn
: is the file name of a file to be brought into virtual storage. If the file is already in storage, no new copy is brought in.

ft
: is the file type of the above file. If you omit ft, the editor uses the file type of the file you are currently editing.

fm
: is the file mode of the file to be edited, indicating an accessed minidisk or SFS directory where the file resides. The editor determines the file mode of the edited file as follows:

- Editing existing files.

  When the file mode is specified, that disk or directory and its extensions are searched. If the file mode is not specified or is specified as an asterisk (*), all accessed disks and/or directories are searched for the specified file.

- Creating new files.

  If the file mode is not specified, the editor assumes a file mode of A1.

options
: are the same as the options in the CMS command XEDIT. (See "Chapter 2: The XEDIT Command.")

Usage Notes

1. The XEDIT subcommand allows you to independently edit multiple files in virtual storage. To edit another file in addition to the one(s) you are already editing, use the XEDIT subcommand and specify the new file ID.

   The files are placed in a *ring*. Each time the XEDIT subcommand is issued without arguments, the next file in the ring appears on the screen as the current file. This arrangement allows you to switch from the first file to the second, the second to the third, etc., all the way around the ring and back to the first.

   Each time the XEDIT subcommand is issued with a file ID, the file name and file type specified are compared to all files in the ring to determine if this file is already being edited. If a match is found, that file is made the current file and any options specified are ignored. If a match is not found, the specified file is brought into storage and becomes the current one.

   A QUIT or FILE subcommand issued from a file causes the previous file in the ring to be displayed.

2. XEDIT in update mode creates a file with a file type of either UPDATE or the file type of the last update file applied, if any. The resulting file may have a file ID identical to that of a file already in the ring. XEDIT in member mode creates a file with file type MEMBER; this file may also have a file ID identical to that of a file already in the ring.

3. When the screen is divided into multiple logical screens (see the SET SCREEN subcommand), entering the XEDIT subcommand with the same file ID from two (or more) logical screens creates two (or more) independent views of the same file.

4. The file ID operands (fn ft fm) each may be specified with an equal (=) sign, in which case the value is the same as that in the current file.

5. If this subcommand is issued as "CMS XEDIT. . .," XEDIT will be called recursively. (See the Usage Notes section of "Chapter 2: The XEDIT Command,.")

Messages

002E	File *fn ft fm* not found [RC=28]
003E	Invalid option: *option* [RC=24]
024E	File XEDTEMP CMSUT1 A1 already exists [RC=28]
029E	Invalid parameter *parameter* in the option *option* field [RC=24]
048E	Invalid mode *mode* [RC=24]
054E	Incomplete fileid specified [RC=24]
062E	Invalid character in fileid *fn ft fm* [RC=20]
065E	*option* option specified twice [RC=24]
066E	*option1* and *option2* are conflicting options [RC=24]
069E	Filemode *mode* not accessed [RC=36]
070E	Invalid parameter *parameter* [RC=24]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 55, or 100]
109S	Virtual storage capacity exceeded [RC=104]
132S	File *fn ft fm* too large [RC=88]
137S	Error *nn* on STATE for *fn ft fm* [RC=88]
229E	Unsupported OS dataset, error *nn* [RC=80, 81, 82, or 83]
500E	Unable to unpack file *fn ft fm* [RC=88]
508E	LOAD must be the first subcommand in the profile [RC=3]
554E	Not enough virtual storage available [RC=104]
555E	File *fn ft fm* already in storage [RC=4]
571I	Creating new file:
622E	Insufficient free storage for {MSGLINE|PFkey/PAkey|synonyms}
915E	Maximum number of windows already defined [RC=13]
927E	The virtual screen must contain at least 5 lines and 20 columns [RC=24]
928E	Command is not valid for virtual screen *CMS* [RC=12]
1138E	File sharing conflict for file *fn ft fm* [RC=70]
1214W	File *fn ft fm* already locked SHARE
1215E	File *fn ft fm* is locked {SHARE|UPDATE|EXCLUSIVE} by another user [RC=70]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 70, 76, 99, or 100]
1299W	Warning: Not authorized to lock file *fn ft fm*
1300E	Error *nn* {locking|unlocking} file *fn ft* {*fm*|*dirname*} [RC=55, 70, 76, 99, or 100]

Messages with Member Option

007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
033E	File *fn ft fm* is not a library [RC=32]
039E	No entries in library *fn ft fm* [RC=32]
167S	Previous MACLIB function not finished [RC=88]
622E	Insufficient free storage for reading map [RC=104]

Messages with Update Options

007E	File *fn ft fm* is not fixed, 80-character records [RC=32]
007E	File *fn ft fm* does not have a logical record length greater than or equal to 80 [RC=32]
007E	File *fn ft fm* does not have the same format and record length as *fn ft fm* [RC=32]
007E	File *fn ft fm* is not fixed record format [RC=32]
104S	Error *nn* reading file *fn ft fm* from disk [RC=31, 32, or 55]
174W	Sequence error introduced in output file: *seqno1* to *seqno2* [RC=32]
178I	Applying *fn ft fm*
179E	Missing or invalid MACS card in control file *fn ft fm*
180W	Missing PTF file *fn ft fm*
183E	Invalid {CONTROL|AUX} file control card [RC=32]
184W	./ S not first card in update file--ignored [RC=32]
185W	Non numeric character in sequence field *seqno* [RC=32]
186W	Sequence number [*seqno1*] not found [RC=32]
207W	Invalid update file control card [RC=32]
210W	Input file sequence error: *seqno1* to *seqno2* [RC=32]
570W	Update *ft* specified in the UNTIL option field not found
597E	Unable to merge updates containing ./S cards [RC=32]
1262S	Error *nn* opening file *fn ft fm* [RC=31, 55, 70, 76, 99, or 100]

Return Codes

0	Normal
4	File is already in storage
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Command is not valid for virtual screen
13	Maximum number of windows already defined
20	Invalid character in file name or file type
24	Invalid parameters or options
28	Source file not found (UPDATE MODE), or library not found (MEMBER option), or specified PROFILE macro does not exist, or file XEDTEMP CMSUT1 already exists
31	A rollback occurred
32	Error during updating process, or file is not a library, or library has no entries, or file is not fixed, 80 char. records
36	Corresponding minidisk or directory not accessed
55	APPC/VM communications error
70	File sharing conflict
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released.
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on.

82	The OS data set or DOS file is not BPAM, BSAM, or QSAM.
83	The OS data set or DOS file has more than 16 user labels or data extents.
88	File is too large and does not fit into storage, or previous Maclib function was not finished
99	A required system resource is not available
100	Error reading the file into storage
104	Insufficient storage available

& (Ampersand)

Use an ampersand (&) at the command line before any subcommand to cause the subcommand to be redisplayed.

Format

&	[*subcommand*]

Usage Notes

1. Using an & allows you to reexecute the subcommand just by pressing the ENTER key. You can also modify the command before reentering it.
2. A synonym cannot be defined for &.
3. The & subcommand must not be preceded by blanks. Also, & will cause trailing blanks in the subcommand that follows it to be replaced with nulls.
4. MSGMODE must be ON in order for & to work properly.

= (Equal Sign)

Use the = subcommand to reexecute the last subcommand or macro or CP/CMS command entered, or to execute a specified subcommand and *then* reexecute the last one entered.

Format

=	[*subcommand*]

Operand

subcommand
is any XEDIT subcommand (or any CP or CMS command, if SET IMPCMSCP ON is in effect). It is executed *before* the previous subcommand is reexecuted.

Usage Notes

1. Multiple adjacent = subcommands (= = = =) in the command line will cause the last subcommand to be executed as many times as there are equal signs specified.
2. The last subcommand that is being reexecuted could have been entered from the command input area on the terminal, from the console stack (via a macro), or from any of the key settings (PF, PA, or ENTER keys).
3. The editor keeps a copy of the last subcommand or macro in an *equal buffer*. The contents of this buffer may be:
   a. Displayed without being changed by using QUERY =
   b. Returned by using the EXTRACT subcommand (EXTRACT/=/)
   c. Specified by using SET = *string*, where string becomes the new content of the equal buffer.
4. The = subcommand may be renamed by using the SYNONYM subcommand.
5. The editor assigns the = subcommand (with no operand) to the PF9 key. When the = subcommand is assigned to a PF or PA key, the default is ONLY = (see SET PF, SET PA).
6. Entering the = subcommand in the format:

   ```
   = subcommand
   ```

   is useful if, for example, you enter a data line on a typewriter terminal while you are in command mode. Instead of switching to input mode and retyping the data line, you can use the following subcommand:

   ```
   = input
   ```
7. When the ENTER key is hit and nothing has been entered on the command line, the ENTER key definition, if any, is placed in the "equal buffer."
8. The results of the execution of the = (Equal Sign) subcommand may not always be the same as that of the execution of the combination of the ? (Question Mark) subcommand and the ENTER key. The execution of certain defined PF keys, prefix subcommands, etc., can result in this difference.

? (Question Mark)

Use the ? subcommand to display in the command area the last XEDIT subcommand (except for an = or ? subcommand), macro, or CP/CMS command executed from the command line. If none of these have been issued, then the ? subcommand will display the original CMS XEDIT command.

Format

?	

Usage Notes

1. The subcommand that is displayed as a result of a ? can be reexecuted by pressing the ENTER key. You can also modify the command before reentering it.
2. Successive execution of ? subcommands will display the previous subcommands. (The previous subcommands are maintained in a ring.)
3. A synonym cannot be defined for the ? subcommand.
4. The ? subcommand can be assigned to a PF or PA key. For example, the editor assigns the ? subcommand to the PF6 key and PA3 key (initial settings). When the ? subcommand is assigned to a PF or PA key, the default is ONLY ? (see SET PF, SET PA).
5. Anything following a ? is ignored except another ?. Multiple ?s can be specified to retrieve previous subcommands. For example, ??? displays the third previous subcommand.
6. Multiple subcommands ending with ? can be re-executed from the command line by pressing the ENTER key. For example, DEL#NEXT#? can be used repeatedly to delete every other line from part or all of a file.
7. The results of the execution of the = (Equal Sign) subcommand may not always be the same as that of the execution of the combination of the ? (Question Mark) subcommand and the ENTER key. The execution of certain defined PF keys, prefix subcommands, etc., can result in this difference.
8. The ? subcommand must not be preceded by blanks. Trailing blanks in the subcommand that is displayed as a result of a ? subcommand will be respected; they are not replaced with nulls.
9. The ? subcommand has no effect if it is issued when the CMDLINE setting is OFF or if it is issued when the console stack is not empty.

Chapter 4. Prefix Subcommands and Macros

Prefix subcommands and macros are "line" subcommands and macros, which are entered by typing over the five-position prefix area. You can use the prefix subcommands and macros to:

- Insert and delete lines
- Continually add successive lines
- Copy, move, and duplicate lines
- Extend the length of a line
- Move the current line pointer
- Display the scale on a particular line
- Display the tab settings on a particular line
- Assign a symbolic name to a line
- Shift lines left or right
- Exclude lines from display
- Redisplay (show) excluded lines.

The prefix subcommands and macros are:

`A`	Add (equivalent to I)
`C`	Copy
`D`	Delete
`E`	Extend
`F`	Following
`I`	Insert
`M`	Move
`P`	Preceding
`S`	Show excluded lines
`SCALE`	Display scale
`SI`	Add successive lines
`TABL`	Display tab line
`X`	Exclude lines
`.xxxx`	Assign symbolic name
`<`	Shift left
`/`	Set current line
`>`	Shift right
`"`	Duplicate.

General Usage Notes

1. Use the subcommands

   ```
   set prefix on right
     or  set prefix on left
   ```

   to display the five-position prefix area (= = = = =).

   You can use SET PREFIX NULLS to display nulls in the prefix area.

2. Prefix subcommands and macros can be typed over any position of the five-character prefix area.

   For example:

   ```
   ====a  (adds a line)
   a====  (adds a line)
   ```

==d==	(deletes a line)
5a===	(adds 5 lines)
3 a==	(adds 3 lines)

are valid ways to enter prefix subcommands. You can type multiple prefix subcommands and macros before pressing the ENTER key.

3. The prefix area is decoded in the following manner:

   - The prefix area is scanned to determine what has been entered. The scan starts at the right, looking for the first character that is different from the original prefix. If it finds one, it then scans starting at the left (as long as some of the prefix area has not been scanned yet), looking for the leftmost character that is different from the original prefix.
   - Any number is taken as an operand.
   - If the name starts with a letter, the name is decoded up to a non-alphabetic character.
   - If the name starts with a non-alphabetic character (a delimiter), the name is decoded up to a blank or an alphabetic character.
   - Whatever follows is taken as an operand.

   For example:

PREFIX	NAME	OP1	OP2	OP3
4a3 5	a	4	3	5
$XXX	$	XXX		
>>5	>>	5		
5a>b	a	5	>b	
5>ab	>	5	ab	

4. If line numbers are displayed in the prefix area (via the SET NUMBER ON subcommand), use prefix subcommands and macros carefully. Some numbers that you type in the prefix area may be ignored. Only the characters that you type over *and* that are different from the numbers in the prefix area are interpreted as the prefix subcommand or macro.

   For example:

The prefix area contains	12345
You type "3a":	123a5
The prefix subcommand is "a".	
The prefix area contains	12345
You type "a4":	1a445
The prefix subcommand is "a4".	
If you typed "a44":	1a445
or if you typed "a445":	1a445

   the editor would still interpret the prefix subcommand as being "a4," because the characters you type over must be different from the ones that are in the line number. If you want "a44" to be recognized in this situation, type "a44" followed by a blank.

5. When using SET PREFIX NULLS with SET NUMBER ON, use prefix subcommands and macros carefully. Some numbers that you type in the prefix area may be ignored.

For example:

The prefix area contains	224
You type "224a":	224a4
The prefix subcommand is "a4".	

The prefix area contains	224
You type "224a":	224a
The prefix subcommand is "a".	

6. When you are editing a file on multiple screens (multiple views of the same file) prefix subcommands and macros may be specified on all of the views.

   If a prefix subcommand or macro is entered in more than one view of the file, but for the same file line, only the prefix subcommand or macro that is lowest and furthest to the right of the virtual screen will be executed.

7. If a prefix subcommand or macro is entered incorrectly, it is displayed in the prefix area, prefixed by a question mark (?). For example, if XX3 were entered (an invalid form of the X prefix macro), the prefix area would display ?XX3.

   If a prefix subcommand or macro causes a pending condition, or if the user entered an unknown prefix subcommand, the following pending notice is displayed in the status area:

   ```
   'value' pending
   ```

   where value is the name of the subcommand or macro that was entered.

   When XEDIT processes prefix subcommands and macros, it checks for syntax errors. If an error is found, the message is displayed in the screen it was entered in and the prefix subcommand is redisplayed, preceded by a ? in the screen in which it was entered. When the pending list is executed, if the prefix subcommand was entered on an invalid line, an error message will be displayed in the screen in which the pending list was executed and the prefix subcommand will be redisplayed on the line in which it was entered, preceded by a ?.

   Prefix macros are processed when the pending list is being executed. If an error is found in the prefix macro, the message will be displayed in the screen in which the pending list is executed. If the prefix macro indicates that the pending entry that was in error should be redisplayed (using SET PENDING ERROR), then that entry will appear in the same screen in which the error message appears.

8. The RESET subcommand (entered on the command line) can be used to cancel any pending prefix subcommands or macros, that is, prefix subcommands or macros that have caused a pending notice to be displayed in the status area.

   Prefix subcommands and macros are executed before any subcommands executed by pressing a PA/PF key, the ENTER key, or before any subcommands that may be typed in the command line, including RESET.

9. If you type a prefix subcommand or macro to delete, copy, or move a number of lines, and the number (n) you specify is greater than the number of lines left in the file, the number (n) is adjusted automatically to the number of lines left in the file.

10. The prefix subcommands and macros for a file are executed starting at the top of the file moving through to the end of the file. Block commands are executed when the end of the block is reached. Prefix subcommands or macros that cannot be executed or are not recognized are left pending. If the following prefix subcommands were entered, D would execute first, then M and F, followed by DD, which will be left pending because there is no matching entry. The numbers in parentheses indicate the order of processing.

    ```
    MM (2)
    D  (1)
    MM (2)
    DD (3)
    F  (2)
    ```

11. When multiple prefix subcommands and/or macros are issued, the cursor is positioned according to the one with the highest priority (see the CURSOR subcommand).

    The following is a list of the priorities associated with the prefix subcommands and macros (along with the priority assigned to changes on the screen and the ENTER key).

`SI`	Priority = 70
`E`	Priority = 60
`A,I`	Priority = 60
`/`	Priority = 50
`"`	Priority = 40
`M`	Priority = 30
`C`	Priority = 30
`S`	Priority = 30
`X`	Priority = 30
`<,>`	Priority = 30
`ENTER key`	Priority = 30
`Screen change`	Priority = 20
`D`	Priority = 10

For example, if both an A and a ″ prefix subcommand are typed and the ENTER key is pressed, the cursor is positioned on the screen where the new line was added by the A prefix subcommand (regardless of whether the ″ preceded or followed the A on the screen).

If M and F prefix subcommands and a > prefix macro are typed and the ENTER key is pressed, the cursor is positioned either on the moved line or on the shifted line, depending on which was specified first on the screen.

Notes for Macro Writers

1. For information on writing prefix macros, see the *VM/SP System Product Editor User's Guide*. See also: SET/QUERY/EXTRACT PENDING and SET/QUERY/EXTRACT PREFIX in this book.
2. By using the CURSOR subcommand, user-written prefix macros can specify where the cursor is to be positioned as well as a priority for this cursor movement. The cursor is positioned at the location specified that has the highest priority when all pending prefix subcommands and macros are executed.
3. Synonyms can be assigned for prefix macros by using the SET PREFIX subcommand with the SYNONYM option. See the *VM/SP System Product Editor User's Guide* for examples of using the SET PREFIX subcommand to

assign synonyms for prefix macros that you write. SET PREFIX subcommands used to assign synonyms can be entered in a user's PROFILE XEDIT file.

The synonyms assigned to the XEDIT prefix macros are as follows:

Macro Synonym(s)	File Identifier
X, XX	PREFIXX XEDIT
S	PRFSHOW XEDIT
<, >, >>, <<	PRFSHIFT XEDIT
.....	SI XEDIT

A (ADD)

Use the A prefix subcommand to add one or more lines immediately following the line in which the A prefix subcommand is entered.

Format

A	-	**add one line**
*n*A	-	**add *n* lines**
A*n*	-	**add *n* lines**

Usage Note

The A prefix subcommand is equivalent to the I prefix subcommand.

Responses

By default, the prefix area on each line that is added is highlighted. For more information, refer to SET COLOR PENDING.

Each line that is added is pre-filled with the current mask (see SET MASK).

If SET IMAGE ON is in effect, the cursor is placed in the first tab column of the first line that was added. Otherwise, it is placed in column 1.

Messages

529E	Subcommand is only valid in {display|editing} mode [RC=3]
557S	No more storage to insert lines [RC=4]
659E	Invalid prefix subcommand: *prefix*

Return Codes

3	Terminal is not a display terminal
4	Insufficient storage available

Example

Refer to the "Examples" section of the D prefix subcommand.

C (COPY)

Use the C prefix subcommand to copy one line, a specified number of lines, or a block of lines. The F (Following) or P (Preceding) prefix subcommands must be used to indicate the destination of the copied line(s).

Format

C	-	**copy line**
C*n*	-	**copy *n* lines**
***n*C**	-	**copy *n* lines**
CC	-	**copy block of lines**

Usage Notes

1. Whenever you enter a C prefix subcommand, you also must enter an F or a P prefix subcommand in the prefix area of another line to indicate the destination of the copied line(s). The destination line and the line(s) to be copied can be on different screen displays but must be within the same file.
2. To copy one line, enter the character C in the prefix area of the line.
3. To copy more than one line, enter C*n* or *n*C in the prefix area of the *first* line of *n* lines to be copied.
4. To copy a block of lines, enter CC in the prefix areas of both the *first* and *last* lines, which can be on different screens.

Responses

The cursor is placed on the first line that was copied (at its new location) when that line is displayed on the resulting screen. If the first copied line was not displayed on the resulting screen, then the cursor will be positioned on the command line.

The notice

```
'C' pending...
```

is displayed in the status area until the required prefix subcommands have been entered, for example, when a C has been entered but an F or a P has not yet been entered. The "pending" status allows you to scroll through the file before entering, for example, the destination line.

When a CC has been entered on only one line of a block, the following message is displayed in the status area:

```
'CC' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

Messages

557S	No more storage to insert lines [RC=4]
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

Return Codes

4	Insufficient storage available

D (DELETE)

Use the D prefix subcommand to delete one line, a specified number of lines, or a block of lines.

Format

D	-	**Delete one line**
D*n*	-	**Delete *n* lines**
***n*D**	-	**Delete *n* lines**
DD	-	**Delete block of lines**

Usage Note

To delete a block of lines, enter DD in the prefix areas of both the first and last lines of the block to be deleted. The beginning and end of the block can be on different screens but must be within the same file.

Responses

When a DD has been entered on only one line of a block of lines to be deleted, the following notice is displayed in the status area:

```
'DD' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

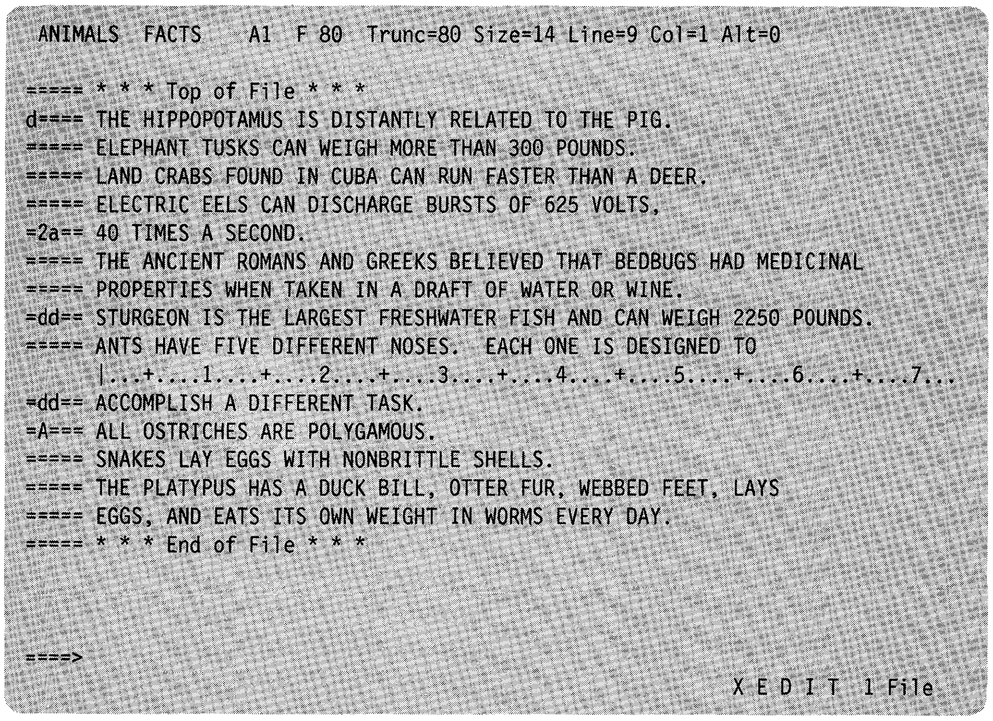
The cursor is placed on the command line. If you want the cursor to be positioned on the line following the last line deleted, you must change the priority of the ENTER (or PA/PF) key.

Messages

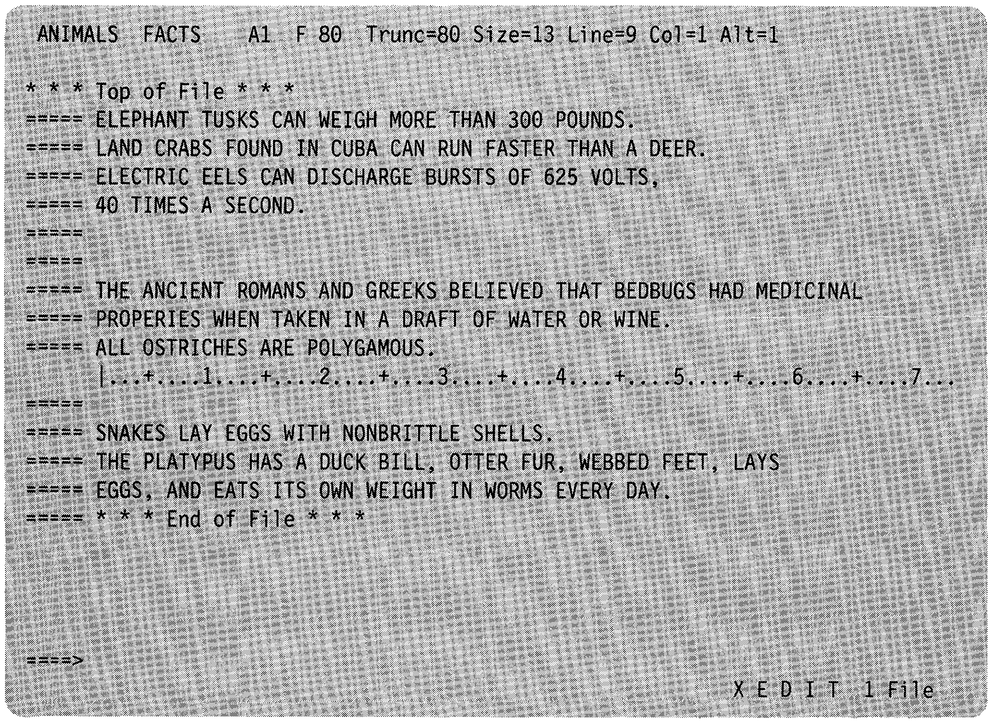
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

Examples

Figure 18 is a before-and-after example of the A and D prefix subcommands.



```
 ANIMALS  FACTS    A1  F 80  Trunc=80 Size=14 Line=9 Col=1 Alt=0

===== * * * Top of File * * *
d==== THE HIPPOPOTAMUS IS DISTANTLY RELATED TO THE PIG.
===== ELEPHANT TUSKS CAN WEIGH MORE THAN 300 POUNDS.
===== LAND CRABS FOUND IN CUBA CAN RUN FASTER THAN A DEER.
===== ELECTRIC EELS CAN DISCHARGE BURSTS OF 625 VOLTS,
=2a== 40 TIMES A SECOND.
===== THE ANCIENT ROMANS AND GREEKS BELIEVED THAT BEDBUGS HAD MEDICINAL
===== PROPERTIES WHEN TAKEN IN A DRAFT OF WATER OR WINE.
=dd== STURGEON IS THE LARGEST FRESHWATER FISH AND CAN WEIGH 2250 POUNDS.
===== ANTS HAVE FIVE DIFFERENT NOSES.  EACH ONE IS DESIGNED TO
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=dd== ACCOMPLISH A DIFFERENT TASK.
=A=== ALL OSTRICHES ARE POLYGAMOUS.
===== SNAKES LAY EGGS WITH NONBRITTLE SHELLS.
===== THE PLATYPUS HAS A DUCK BILL, OTTER FUR, WEBBED FEET, LAYS
===== EGGS, AND EATS ITS OWN WEIGHT IN WORMS EVERY DAY.
===== * * * End of File * * *



====>
                                                              X E D I T  1 File
```



```
 ANIMALS  FACTS    A1  F 80  Trunc=80 Size=13 Line=9 Col=1 Alt=1

* * * Top of File * * *
===== ELEPHANT TUSKS CAN WEIGH MORE THAN 300 POUNDS.
===== LAND CRABS FOUND IN CUBA CAN RUN FASTER THAN A DEER.
===== ELECTRIC EELS CAN DISCHARGE BURSTS OF 625 VOLTS,
===== 40 TIMES A SECOND.
=====
=====
===== THE ANCIENT ROMANS AND GREEKS BELIEVED THAT BEDBUGS HAD MEDICINAL
===== PROPERIES WHEN TAKEN IN A DRAFT OF WATER OR WINE.
===== ALL OSTRICHES ARE POLYGAMOUS.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
===== SNAKES LAY EGGS WITH NONBRITTLE SHELLS.
===== THE PLATYPUS HAS A DUCK BILL, OTTER FUR, WEBBED FEET, LAYS
===== EGGS, AND EATS ITS OWN WEIGHT IN WORMS EVERY DAY.
===== * * * End of File * * *



====>
                                                              X E D I T  1 File
```

Figure 18. Prefix Subcommands A and D — Before and After

E (EXTEND)

Use the E prefix subcommand to extend a logical line by one more virtual screen line.

Format

```
E
```

Usage Notes

1. After an E prefix subcommand is entered, the entire logical line, which now appears on two virtual screen lines, is treated as one line. Shifting due to character deletion and insertion affects the entire logical line.
2. The entire logical line visible on the screen does not exceed the maximum value defined by the SET VERIFY subcommand.
3. A line cannot be extended on a vertical screen (see SET SCREEN in this book).

Responses

The cursor is positioned following the last non-blank character.

Messages

659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

F (FOLLOWING)

Use the F prefix subcommand to identify the line *after* which lines are to be copied or moved via the C or M prefix subcommands.

Format

```
F
```

Response

The notice

```
'F' pending...
```

is displayed in the status area if an F prefix subcommand has been entered and an associated C or M prefix subcommand has not yet been entered. The "pending" status allows you to scroll through the file before entering a C or M prefix subcommand.

Message

659E Invalid prefix subcommand: *prefix*

Example

See Figure 19 (the M prefix subcommand).

I (INSERT)

Use the I prefix subcommand to insert one or more lines immediately following the line in which the I prefix subcommand is entered.

Format

```
I   -  Insert one line
nI  -  Insert n lines
In  -  Insert n lines
```

Usage Note

The I prefix subcommand is identical to the A prefix subcommand.

Responses

By default, the prefix area of each line that is inserted is highlighted. For more information, refer to SET COLOR PENDING.

Each line that is inserted is pre-filled with the current mask (see the SET MASK subcommand).

If SET IMAGE ON is in effect, the cursor is placed in the first tab column of the first line that was inserted. Otherwise, it is placed in column 1.

Messages

529E	Subcommand is only valid in {display|editing} mode [RC=3]
557S	No more storage to insert lines [RC=4]
659E	Invalid prefix subcommand: *prefix*

Return Codes

3	Terminal is not a display terminal
4	Insufficient storage available

M (MOVE)

Use the M prefix subcommand to move one line, a specified number of lines, or a block of lines from one location to another in the file. The original lines are deleted. The F (Following) or P (Preceding) subcommand must be used to indicate the destination of the lines that are moved.

Format

M	-	**move one line**
M*n*	-	**move *n* lines**
***n*M**	-	**move *n* lines**
MM	-	**move block of lines**

Usage Notes

1. Whenever you enter an M prefix subcommand, you also must enter an F or a P prefix subcommand in the prefix area of another line to indicate the destination of the lines to be moved. The destination line and the line(s) to be moved can be on different screen displays but must be within the same file.
2. To move one line, enter the character M in the prefix area of the line.
3. To move more than one line, enter M*n* or *n*M in the prefix area of the *first* line of *n* lines to be moved.
4. To move a block of lines, enter MM on both the *first* and *last* lines, which can be on different screen displays but must be within the same file.

Responses

The notice

```
'M' pending...
```

is displayed in the status area until the required prefix subcommands have been entered, for example, when an M has been entered but the F or P has not yet been entered. The "pending" status allows you to scroll through the file before you enter another prefix subcommand (the destination line, for example).

When an MM has been entered on only one line of a block, the following notice is displayed in the status area:

```
'MM' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

After the move, the cursor is positioned on the first line that was moved when that line is displayed on the resulting screen. If the first line that was moved is not displayed on the resulting screen, then the cursor will be positioned on the command line.

Messages

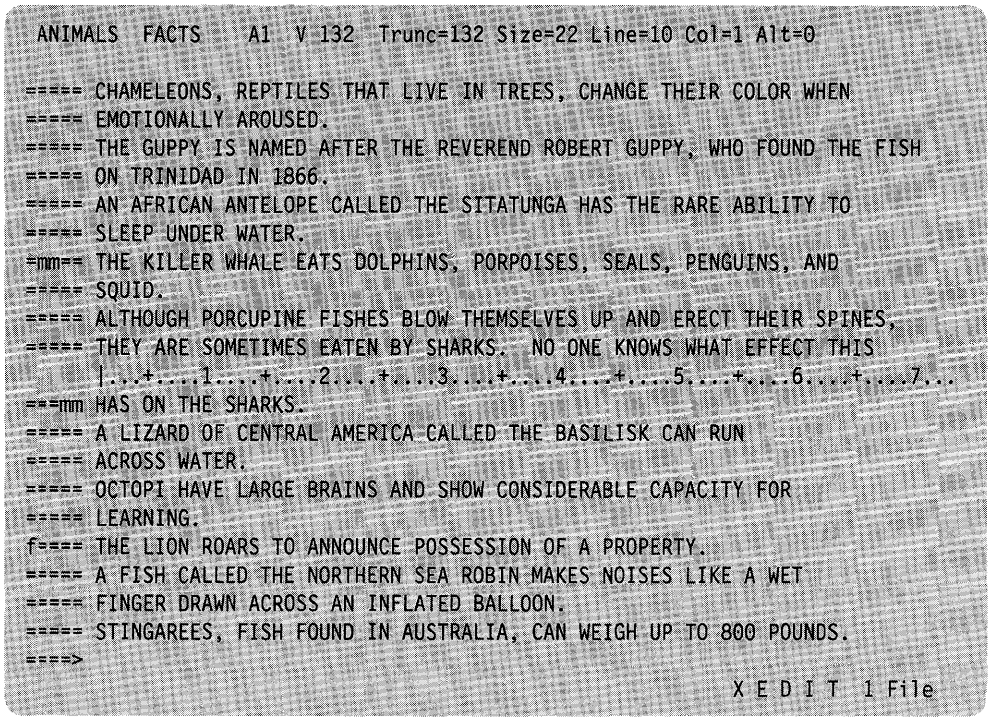
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered
557S	No more storage to insert lines [RC=4]

Return Codes

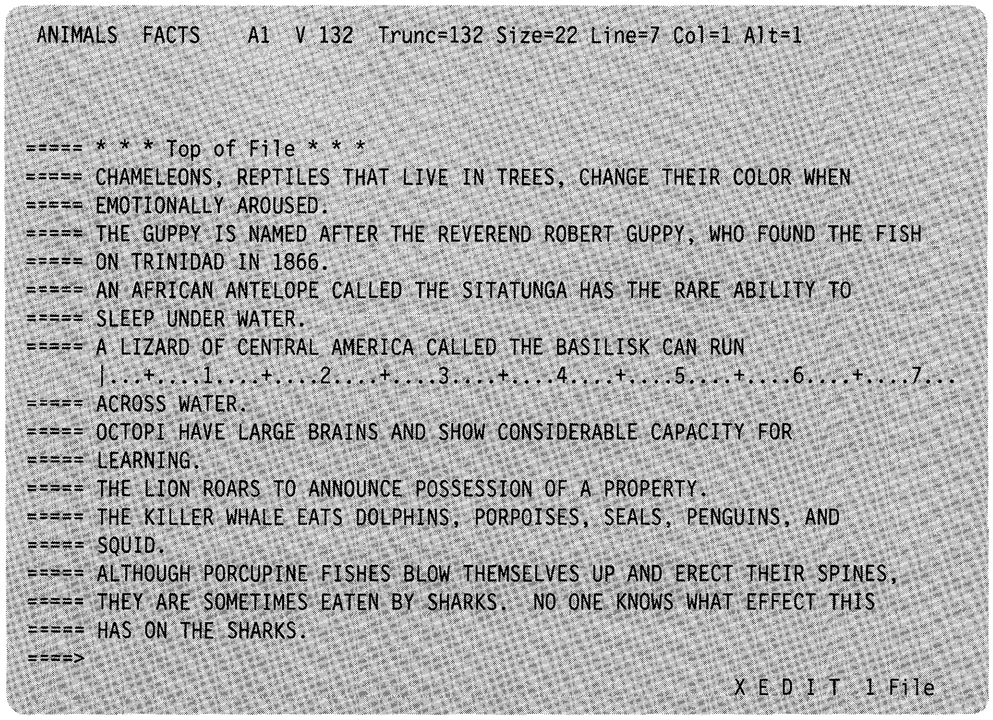
4	Insufficient storage available

Example

Figure 19 is a before-and-after example of the M and F prefix subcommands.



```
 ANIMALS  FACTS    A1  V 132  Trunc=132 Size=22 Line=10 Col=1 Alt=0

===== CHAMELEONS, REPTILES THAT LIVE IN TREES, CHANGE THEIR COLOR WHEN
===== EMOTIONALLY AROUSED.
===== THE GUPPY IS NAMED AFTER THE REVEREND ROBERT GUPPY, WHO FOUND THE FISH
===== ON TRINIDAD IN 1866.
===== AN AFRICAN ANTELOPE CALLED THE SITATUNGA HAS THE RARE ABILITY TO
===== SLEEP UNDER WATER.
=mm== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===mm HAS ON THE SHARKS.
===== A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
f==== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
===== FINGER DRAWN ACROSS AN INFLATED BALLOON.
===== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS.
====>
                                                             X E D I T  1 File
```



```
 ANIMALS  FACTS    A1  V 132  Trunc=132 Size=22 Line=7 Col=1 Alt=1



===== * * * Top of File * * *
===== CHAMELEONS, REPTILES THAT LIVE IN TREES, CHANGE THEIR COLOR WHEN
===== EMOTIONALLY AROUSED.
===== THE GUPPY IS NAMED AFTER THE REVEREND ROBERT GUPPY, WHO FOUND THE FISH
===== ON TRINIDAD IN 1866.
===== AN AFRICAN ANTELOPE CALLED THE SITATUNGA HAS THE RARE ABILITY TO
===== SLEEP UNDER WATER.
===== A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== ACROSS WATER.
===== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
===== LEARNING.
===== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
===== THE KILLER WHALE EATS DOLPHINS, PORPOISES, SEALS, PENGUINS, AND
===== SQUID.
===== ALTHOUGH PORCUPINE FISHES BLOW THEMSELVES UP AND ERECT THEIR SPINES,
===== THEY ARE SOMETIMES EATEN BY SHARKS.  NO ONE KNOWS WHAT EFFECT THIS
===== HAS ON THE SHARKS.
====>
                                                             X E D I T  1 File
```

Figure 19. Prefix Subcommands M and F — Before and After

P (PRECEDING)

Use the P prefix subcommand to identify the line *before* which lines are to be copied or moved via the C or M prefix subcommands.

Format

```
P
```

Response

The notice

```
'P' pending...
```

is displayed in the status area if a P prefix subcommand has been entered and an associated C or M prefix subcommand has not yet been entered. The "pending" status allows you to scroll through the file before entering a C or M prefix subcommand.

Message

659E	Invalid prefix subcommand: *prefix*

S (SHOW) Macro

Use the S prefix macro to redisplay one or more lines that were excluded by the X prefix macro, the ALL macro, or other selective line editing subcommands (SET SELECT or SET DISPLAY). The S prefix macro can be entered only in the prefix area of a shadow line (see SET SHADOW).

Format

S	-	**show all lines**
S*	-	**show all lines**
S*n*	-	**show the first *n* lines**
S+*n*	-	**show the first *n* lines**
***n*S**	-	**show the first *n* lines**
S-*n*	-	**show the last *n* lines**

Usage Notes

1. When n or +n is specified, lines are redisplayed starting at the beginning of the group of excluded lines. When −n is specified, lines are redisplayed starting at the end of the group of excluded lines. (However, they are displayed in ascending order. For example, if lines 1 through 10 are excluded, S−2 redisplays lines 9 and 10, in that order.)

   If +n or −n is larger than the number of excluded lines in the group, n is automatically adjusted to display all of the excluded lines.

2. Redisplayed lines are included in the scope of regular editing subcommands.

3. The S prefix macro alters the selection level (see SET SELECT) of the redisplayed lines to the n2 value of SET DISPLAY n1[n2].

Note for Macro Writers

The file identifier for the S prefix macro is PRFSHOW XEDIT.

Responses

If all lines in a group of excluded lines are redisplayed, the shadow line disappears. If one or more lines remain excluded, the notice in the shadow line is adjusted accordingly. The cursor is placed on the first redisplayed line.

Messages

646E	*macroname* must be invoked from the prefix area [RC=8]
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

Return Codes

8	Subcommand must be issued from prefix area

SCALE (DISPLAY SCALE)

Use the SCALE prefix subcommand to display the scale on the corresponding screen line.

Format

```
SCALE
```

Usage Note

The SCALE prefix subcommand has the same effect as the subcommand:

```
set scale on n
```

Responses

```
The scale looks like this:

<...+..|.1....+....2....+....3.>..+....4T...+....5....+....6....+....7...
.      .                       .     .
.      .                       .     .
.      .column pointer         .     .
.                              .     .truncation column
.left zone                     .right zone
```

Message

659E Invalid prefix subcommand: *prefix*

SI (STRUCTURED INPUT) Macro

Use the SI prefix macro to add a line immediately following the line on which SI was specified and position the cursor at the column where the text on the preceding line begins. Another new line will be added each time that you type on the new line and then press the ENTER key. SI continues to insert new lines until ENTER is pressed without typing on the new line.

Format

SI

Usage Notes

1. SI can also be issued as a subcommand. See the SI macro.
2. Using SI you can add a blank line in a file by making at least one change on the new line. The line is considered changed if you press the space bar or if you retype the mask characters (see SET MASK). Moving the cursor over a line using the cursor position keys does not change the line.
3. After SI is terminated, the cursor is positioned at the indentation column of the last added line.
4. As lines are added using SI, any data above the new line remains stationary while the data below scrolls down the screen. When the new line is one line above the bottom of the file area, adding more lines will scroll the data above the new line up the screen. Using multiple SI commands or other prefix subcommands along with SI may move the new line off the screen. The cursor may not be placed at the indentation column when multiple SI subcommands are issued.

Note for Macro Writers

SI uses the console stack to adjust the current line when anything is pending on the line that will become the current line.

Responses

The prefix area of the new line contains '.....' and the following message is displayed in the status area:

```
'.....' pending...
```

This pending status allows you to continually add lines after you have typed on the new line.

By default, the prefix area for the line that is added is highlighted. For more information, refer to SET COLOR PENDING.

Each line that is added is pre-filled with the current mask (see SET MASK).

Messages

529E	SI is only valid in {display|editing} mode [RC=3]
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

Return Codes

3	Terminal is not a display terminal

Example

To add more ingredients following the line that contains the words "graham cracker crumbs," type SI in the prefix area of that line.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=si==      1 1/2  Cups graham cracker crumbs
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

Pressing the ENTER key results in a new line being added. Note that the prefix area of the new line contains '. . . . .' and that the cursor has been indented on the new line.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
.....      _
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

Type an ingredient on the new line and press the ENTER key.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
.....      8      Ounces sweetened condensed milk _
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

The '. . . . .' notice and the cursor move from the line you just typed on to another new line.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
=====      8      Ounces sweetened condensed milk
.....      _
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

You can continue to enter ingredients by typing on the new line and pressing the ENTER key. When you are finished entering ingredients, then press the ENTER key (in this example the new line was unchanged).

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
=====      8      Ounces sweetened condensed milk
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

TABL (DISPLAY TAB LINE)

Use the TABL prefix subcommand to display a "T" in every tab column, according to the current tab settings (see SET TABS in this book), on the corresponding screen line.

Format

```
TABL
```

Usage Note

The TABL prefix subcommand has the same effect as the subcommand:

```
set tabline on n
```

Responses

The line displays a "T" in every tab column:

For example:

```
T     T     T     T     T     T     T     T     T
```

Message

659E Invalid prefix subcommand: *prefix*

X (EXCLUDE) Macro

Use the X prefix macro to exclude from the display either one line, a specified number of lines, or a block of lines. Lines excluded from the display are also excluded from the scope of editing subcommands (see SET SCOPE).

Format

X	-	**exclude one line from display**
X*n*	-	**exclude *n* lines from display**
***n*X**	-	**exclude *n* lines from display**
XX	-	**exclude a block of lines from display**

Usage Notes

1. To exclude a block of lines, enter XX in the prefix area of both the first and last lines of the block, which can be on different screens but must be within the same file.
2. Use the S (SHOW) prefix macro to redisplay the excluded lines.
3. The lines excluded from display are also excluded from the scope of regular editing subcommands, unless you specify SET SCOPE ALL.
4. The X prefix macro alters the selection level of the lines excluded (see SET SELECT). Excluded lines are assigned a selection level that is one greater than the end of the display range (see SET DISPLAY). If you have entered SET DISPLAY n *, the X prefix macro has no effect. The X prefix macro does not alter the settings of SET SCOPE, SET SHADOW, or SET DISPLAY.
5. The lines excluded are replaced with a "shadow line" (see SET SHADOW), which indicates the number of lines excluded. If you do not want the shadow line displayed, issue SET SHADOW OFF.
6. If a block of lines to be excluded includes lines that were *previously* excluded (that is, these lines are nested within the block), these lines remain excluded. However, their selection level remains the same. The shadow line shows the total number of lines excluded.

Note for Macro Writers

The X prefix macro is an example of how selective line editing subcommands (SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW) can be used. The file identifier for the X prefix macro is PREFIXX XEDIT.

Responses

When XX has been entered on only one line of a block of lines to be excluded and a key is pressed, the XX prefix macro is displayed highlighted in the prefix area, and the status area displays the following pending notice:

```
'XX' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

The cursor is placed on the first line following the excluded line(s).

Messages

646E	*macroname* must be invoked from the prefix area [RC=8]
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered
686E	Synonym *name* not recognized by prefix macro *macroname*

Return Codes

8	Subcommand must be issued from prefix area

.xxxx (SET SYMBOLIC NAME)

Use the .xxxx prefix subcommand to assign a symbolic name to a line. One or more names can be defined for a line using separate .xxxx subcommands. You can use a symbolic name to refer to the line in subsequent target operands of XEDIT subcommands.

Format

```
.xxxx
```

Operand

.xxxx
is a symbolic name for the line. The name must begin with a period and be followed by from one to four alphanumeric characters. For example, .AAA.

Usage Notes

1. The .xxxx prefix subcommand is the same as the SET POINT subcommand, except that .xxxx limits the name to four characters.
2. The .xxxx prefix subcommand makes it unnecessary for you to remember or to look up the line number. A line can be referenced by its name at any time during an editing session.
3. A symbolic name stays with a line for the entire editing session, even if the line number changes due to insertion or deletion of other lines. However, you can delete a symbolic name by using the SET POINT subcommand (SET POINT *.xxxx* OFF). You can also delete a symbolic name for one line by using .xxxx to assign that name to another line.
4. After a symbolic name is defined for a line, the name does *not* appear in the prefix area. You must keep track of symbolic names. The subcommand QUERY POINT * can be used to display all names and their line numbers currently defined for the file. The subcommand QUERY POINT can be used to display the name(s) of the current line.

Message

659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

< (SHIFT LEFT) Macro

Use the < prefix macro to shift one line or a block of lines one or more columns to the left.

Format

<	-	**shift one line one column to the left**
<*n*	-	**shift one line *n* columns to the left**
n <	-	**shift one line *n* columns to the left**
< <	-	**shift a block of lines one column to the left**
< < *n*	-	**shift a block of lines *n* columns to the left**
n < <	-	**shift a block of lines *n* columns to the left**

Usage Notes

1. Data shifted to the left past the zone1 column is lost. The line is padded with blanks to the right, up through the truncation column. (The < prefix macro is comparable to the SHIFT subcommand issued with the LEFT operand.)
2. To shift a block of lines one column to the left, enter < < in the prefix areas of both the first and last lines of the block. The beginning and end of the block can be on different screens but must be within the same file.

   To shift a block of lines n columns to the left, enter < <n or n< < in either the first or last line of the block, and < < in the other. (If you enter a number on both the first and last lines of the block, the one closest to the end of file is used.)

Note for Macro Writers

The file identifier for the < prefix macro is PRFSHIFT XEDIT.

Responses

When < < has been entered on only one line of a block of lines to be shifted and a key is pressed, the < < prefix macro is displayed highlighted in the prefix area, and the status area displays the following pending notice:

```
'<<' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

The cursor is placed on the first line shifted.

Messages

646E	*macroname* must be invoked from the prefix area [RC=8]
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered
686E	Synonym *name* not recognized by prefix macro *macroname*

Return Codes

8	Subcommand must be issued from prefix area

/ (SET CURRENT LINE)

Use the / (diagonal) prefix subcommand to set the current line or to identify a line that will be the new current line after other prefix subcommands are executed and, optionally, to move the column pointer.

Format

```
/[n] or [n]/
```

Operand

n

is the column number in which the column pointer is to be placed.

Usage Notes

1. If several / prefix subcommands are typed on the screen, the last one will set the current line.
2. If SET IMAGE ON is in effect, the cursor is placed in the first tab column of the new current line. Otherwise, it is placed in column 1.

Messages

659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered

> (SHIFT RIGHT) Macro

Use the > prefix macro to shift one line or a block of lines one or more columns to the right.

Format

>	-	**shift one line one column to the right**
>*n*	-	**shift one line *n* columns to the right**
n >	-	**shift one line *n* columns to the right**
> >	-	**shift a block of lines one column to the right**
> > *n*	-	**shift a block of lines *n* columns to the right**
n > >	-	**shift a block of lines *n* columns to the right**

Usage Notes

1. Shifted data that extends past the truncation column is either lost or "spilled" (see SET SPILL). The line is padded to the left with blanks. (The > prefix macro is comparable to the SHIFT subcommand issued with the RIGHT operand.)
2. To shift a block of lines one column to the right, enter > > in the prefix areas of both the first and last lines of the block. The beginning and end of the block can be on different screens but must be within the same file.

   To shift a block of lines n columns to the right, enter > >n or n> > in either the first or last line of the block, and > > in the other. (If you enter a number on both the first and last lines of the block, the one closest to the end of file is used.)

Note for Macro Writers

The file identifier for the > prefix macro is PRFSHIFT XEDIT.

Responses

When > > has been entered on only one line of a block of lines to be shifted and a key is pressed, the > > prefix macro is displayed highlighted in the prefix area, and the status area displays the following pending notice:

```
'>>' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

The cursor is placed on the first line shifted.

Messages

646E	*macroname* must be invoked from the prefix area [RC=8]
659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered
686E	Synonym *name* not recognized by prefix macro *macroname*

Return Codes

8	Subcommand must be issued from prefix area

" (DUPLICATE)

Use the " (double quote) prefix subcommand to duplicate one line or a block of lines, either one time or a specified number of times.

Format

"	-	**duplicate one line**
"*n* or *n*"	-	**duplicate line *n* times**
" "	-	**duplicate block of lines**
" "*n* or *n* " "	-	**duplicate block *n* times**

Usage Notes

1. To duplicate a block of lines, enter "" on both the first and last lines of the block. The beginning and end of the block can be on different screens.
2. To duplicate a block of lines n times, enter ""*n* or *n*"" on the first line of the block, and enter "" on the last line of the block.

Responses

When "" has been entered on only one line of a block, the following notice is displayed in the status area:

```
'""' pending...
```

This "pending" status allows you to scroll through the file before completing the block.

The cursor is positioned on the duplicated line.

Messages

659E	Invalid prefix subcommand: *prefix*
661E	Prefix *name* is invalid for the line on which it was entered
557S	No more storage to insert lines [RC=4]

Return Codes

4	Insufficient storage available

Appendix A. File Type Defaults

FILE TYPE	SERIAL	TRUNC	LRECL	RECFM	VERIFY	ESCAPE	CASE	SPILL	IMAGE
$EXEC	ON	72	80	F	72	/	M	OFF	ON
$XEDIT	ON	72	80	F	72	/	M	OFF	ON
AMSERV	ON	72	80	F	T	/	U	OFF	ON
ASM3705	ON	71	80	F	T	/	U	OFF	ON
ASSEMBLE	ON	71	80	F	72	/	U	OFF	ON
BASDATA	OFF	255	255	V	T	/	M	OFF	ON
BASIC	OFF	156	156	V	T	/	M	OFF	ON
CNTRL	OFF	80	80	F	T	/	U	OFF	ON
COBOL	ON	72	80	F	T	/	U	OFF	ON
COPY	ON	71	80	F	T	/	U	OFF	ON
DLCS	ON	72	80	F	72	/	U	OFF	ON
DIRECT	ON	72	80	F	T	/	U	OFF	ON
ESERV	ON	71	80	F	T	/	U	OFF	ON
EXEC	OFF	130	130	V	T	/	U	OFF	ON
FORTRAN	ON	72	80	F	T	/	U	OFF	ON
FREEFORT	OFF	81	81	V	T	/	U	OFF	ON
GCS	OFF	130	130	V	T	/	U	OFF	ON
GROUP	ON	71	80	F	72	/	U	OFF	ON
JOB	OFF	80	80	F	T	+	U	OFF	ON
LISTING	OFF	121	121	V	T	/	U	OFF	ON
MACLIB	OFF	71	80	F	72	/	U	OFF	OFF
MACRO	ON	71	80	F	72	/	U	OFF	ON
MEMBER	ON	71	80	F	72	/	U	OFF	ON
MEMO	OFF	80	80	V	T	/	M	WORD	ON
MODULE	OFF	80	80	V	72	/	M	OFF	OFF
NAMES	OFF	255	255	V	T	/	M	OFF	ON
NETLOG	OFF	255	255	V	T	/	M	OFF	ON
NOTE	OFF	132	132	V	T	/	M	WORD	ON
NOTEBOOK	OFF	132	132	V	T	/	M	WORD	ON
PASCAL	OFF	72	72	V	T	/	M	OFF	ON
PLI	ON	72	80	F	T	/	U	OFF	ON
PLIOPT	ON	72	80	F	T	/	U	OFF	ON
SCRIPT	OFF	132	132	V	T	/	M	WORD	CANON
TEXT	OFF	80	80	F	72	/	M	OFF	OFF
UPDATE	ON	71	80	F	72	/	U	OFF	ON
UPDT	ON	71	80	F	72	/	U	OFF	ON
VSBASIC	OFF	80	80	F	T	/	U	OFF	ON
VSBDATA	OFF	132	132	V	T	/	U	OFF	ON
XEDIT	OFF	255	255	V	T	/	U	OFF	ON
Other	OFF	80	80	F	**	/	U	OFF	ON

Where VERIFY = T means verify = trunc column and Verify = ** means verify = screen size.

FILE TYPE	Tab Settings
$EXEC	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
$XEDIT	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
AMSERV	2 5 10 15 20 25 30 35 40 45 50 55 60
ASM3705	1 10 16 30 35 40 45 50 55 60 65 70
ASSEMBLE	1 10 16 30 35 40 45 50 55 60 65 70
BASDATA	1 7 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
BASIC	1 7 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
CNTRL	1 5 8 17 27 31
COBOL	1 8 12 20 28 36 44 68 72 80
COPY	1 10 16 30 35 40 45 50 55 60 65 70
DLCS	1 4 7 10 13 16 19 22 25 31 37 43 49 55
DIRECT	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
ESERV	2 5 10 15 20 25 30 35 40 45 50 55 60
EXEC	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
FORTRAN	1 7 10 15 20 25 30 80
FREEFORT	9 15 18 23 28 33 38 81
GCS	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
GROUP	1 10 16 30 35 40 45 50 55 60 65 70
JOB	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
LISTING	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
MACLIB	1 10 16 30 35 40 45 50 55 60 65 70
MACRO	1 10 16 30 35 40 45 50 55 60 65 70
MEMBER	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
MEMO	1 10 16 30 35 40 45 50 55 60 65 70
MODULE	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NAMES	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NETLOG	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NOTE	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NOTEBOOK	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
PASCAL	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
PLI	2 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
PLIOPT	2 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
SCRIPT	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
TEXT	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
UPDATE	1 10 16 30 35 40 45 50 55 60 65 70
UPDT	1 10 16 30 35 40 45 50 55 60 65 70
VSBASIC	7 10 15 20 25 30 80
VSBDATA	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
XEDIT	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
Other	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120

Appendix B. Effects of Selective Line Editing Subcommands

SELECTIVE LINE EDITING - A GENERAL DESCRIPTION

Selective line editing can be used in a macro to control both the action of the editor and the screen display. It consists of four subcommands: SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW, which work together in the following manner. You can use SET SELECT to assign a "selection level," or value, to one or more lines in a file. Lines can be logically grouped by assigning them the same selection level. SET DISPLAY may be used in conjunction with SET SELECT to display those lines which have the same selection level. SET SCOPE defines the set of lines that the editor can act upon. SCOPE DISPLAY is the initial setting and, as such, restricts editor action to only those lines that are defined by SET DISPLAY. By default, SET SHADOW displays a notice indicating how many lines are not being displayed in the physical position of the excluded lines in the file. If SET SHADOW is "OFF," only those lines defined by SET DISPLAY will appear on the screen, with no shadow lines to indicate where lines are not being displayed.

Some subcommands automatically cause specific "SETs" to be made within the file when they are invoked from a selective line editing environment. This appendix addresses those "automatic" sets, in addition to differences in subcommand operation when invoked from different selective line editing environments.

AUTOMATIC SELECTION LEVEL ASSIGNMENT: The initial selection level for all lines in a file is 0. When new lines are being added to a file, they are automatically assigned a selection level. That selection level is dependent upon the subcommand that was used to add the new lines. Below is a list of those subcommands and how each affects selection level assignment.

ADD	New line has selection level of n1 (the first selection level defined for inclusion in the screen display) of SET DISPLAY n1 n2. (Also applies to A/I prefix subcommands).
COPY	Copied lines have the same selection level(s) as they do in their original position in the file. (Also applies to C prefix subcommand).
DUPLICAT	Duplicated lines have the same selection level(s) as they originally had, prior to duplication. (Also applies to prefix subcommand).
GET	Lines inserted in the file by the usage of the GET subcommand have selection level n1 (the first selection level defined for inclusion in the screen display) of SET DISPLAY n1 n2.
INPUT	New line has selection level of n1 (the first selection level defined for inclusion in the screen display) of SET DISPLAY n1 n2.
JOIN	Joined lines have the same selection level as the original set of lines with which they are being joined.
MERGE	Merged lines have the same selection level as the lines with which they are being merged.
MOVE	Moved lines retain their original selection level(s). (Also applies to M prefix subcommand).
RECOVER	Recovered line has a selection level of n1 (the first selection level defined for inclusion in the screen display) of SET DISPLAY n1 n2.

REPLACE	When entering the REPLACE subcommand with text, the new line has the same selection level as the line it is replacing. Entering the REPLACE subcommand without text will delete the current line and cause you to enter input mode. The selection level of the new lines inserted while you are in input mode is n1 of the SET DISPLAY n1 n2.
SET SPILL	New line(s) that is created as a result of being spilled has selection level n1 (the first selection level defined for inclusion in the screen display) of SET DISPLAY n1 n2.
SPLIT	New line(s) created by a split have the same selection level(s) as the original line.

HOW THE SCOPE SETTING AFFECTS THE ACTION OF SOME SUBCOMMANDS: The action of some subcommands is also dependent upon the SCOPE setting. These subcommands can be divided into two functional groups:

1. Those which perform target processing or searches and
2. Those which perform operations on file lines.

Target Processing and Searches: Target searches are performed only on lines within the current scope. If SCOPE DISPLAY has been set, the target search only considers and looks at displayed lines. This is true for all types of targets: absolute line numbers, a relative displacement from the current line, a line name, a simple string expression, or a complex string expression. Line movement by use of targets is done as if lines outside the scope had been removed from the file. For example, NEXT or +1 may go from line 20 to line 40 if lines 21 to 39 are outside the display range. If SCOPE ALL has been set, the editor acts on the entire file.

Subcommand	Scope Display	Scope All
BOTTOM	The last line of the scope becomes the current line. (Line must be displayed.)	The last line of the file becomes the current line. (Line may or may not be displayed.)
CLOCATE /string/	String must be in the scope to be located. (Line containing string must be displayed.)	String must be in the file to be located. (Line containing string may or may not be displayed.)
DOWN, NEXT	Next line in the scope becomes the current line. (Line must be displayed.)	Next line in the file becomes the current line. (Line may or may not be displayed.)
FIND or FINDUP	Searches forward or backward in the file for the first line within the scope that starts with the text specified in the operand. (Line must be displayed.)	Searches forward or backward in the file for the first line that starts with the text specified in the operand. (Line may or may not be displayed.)
LOCATE +1	Next line in the scope is located. (Line must be displayed.)	Next line in the file is located. (Line may or may not be displayed.)
NFIND or NFINDUP	Searches forward or backward in the file for the first line within the scope that does not start with the text specified in the operand. (Line must be displayed.)	Searches forward or backward in the file for the first line that does not start with the text specified in the operand. (Line may or may not be displayed.)

Subcommand	Scope Display	Scope All
`UP`	Previous line in the scope becomes the current line. (Line must be displayed.)	Previous line in the file becomes the current line. (Line may or may not be displayed.)

This concept applies for every subcommand that takes a target except SET RANGE, SORT, and the macro ALL, which are special cases that operate outside of the SCOPE. They execute as if SCOPE ALL were in effect, no matter what the SCOPE setting is. For example:

Subcommand	Scope Display	Scope All
`SET RANGE :1 :20`	RANGE is set from line 1 to 20 whether 1 and 20 are in the scope or not. (Lines 1 and 20 may not be displayed.)	Same as SCOPE DISPLAY
`SORT /NY/ 1 5`	Columns 1 to 5 are sorted into ascending sequence for all lines in the file from the current line up to but not including the line containing the string NY. (Lines may or may not be displayed.)	Same as SCOPE DISPLAY
`ALL /SET/`	All lines in the file with string SET are selected for editing (Lines may or may not be displayed.)	Same as SCOPE DISPLAY

Operations: The operation of many subcommands is affected by the SCOPE setting that has been defined. A few examples follow to illustrate how some of those operations are affected.

Subcommand	Scope Display	Scope All
`CHANGE/Record/Line/*`	"Record" is changed to "Line" in all lines in the scope from the current line until the end of the file. (Lines must be displayed.)	"Record" is changed to "Line" in all lines in the file from the current line until the end of the file. (Lines may or may not be displayed.)
`MOVE 3 /DATA/`	Three lines that are in the scope starting with the current line are moved after the line containing the string DATA. (The three lines must be displayed.)	Three lines that are in the file starting with the current line are moved after the line containing the string DATA. (The three lines may or may not be displayed.)
`UPPERCAS 20`	20 lines starting from the current line that are in the scope are translated to uppercase. (Lines must be displayed.)	20 lines from the current line that are in the file are translated to uppercase. (Lines may or may not be displayed.)

This concept applies for every subcommand that performs an operation including prefix subcommands and macros, except SORT and the macro ALL. (See previous description.)

ISSUING PREFIX SUBCOMMANDS AND MACROS FROM A SHADOW LINE: If SCOPE DISPLAY is in effect (the default), all prefix subcommands and macros are invalid if entered on a shadow line with the following exceptions:

Subcommand	Scope Display	Scope All
A	Line is added after the last line represented by the shadow line.	Line is added after the first line represented by the shadow line.
F	Line(s) will be moved or copied after the last line represented by the shadow line.	Line(s) will be moved or copied after the first line represented by the shadow line.
I	Line is inserted after the last line represented by the shadow line.	Line is inserted after the first line represented by the shadow line.
P	Line(s) will be moved or copied before the first line represented by the shadow line.	Line(s) will be moved or copied before the first line represented by the shadow line.
S	Line(s) will be redisplayed.	Line(s) will be redisplayed.

If SCOPE ALL is in effect, all prefix subcommands and macros can be entered on a shadow line except SI. The operation requested is performed on file lines whether they are displayed or not. For example, entering D3 on a shadow line will delete the next three lines in the file whether these lines are represented by a shadow line or are displayed.

For prefix subcommands and macros with block operations, shadow lines (excluded lines) within the block are handled differently depending on the SCOPE setting. See the example below.

Example: In the following segment of a file, a shadow line falls within a block prefix subcommand entry.

```
===== The resort is comprised of 500 acres.
===== It has a private pond, waterfalls, brooks, and woodlands.
===== It is just one step away from hiking, swimming, and skating.
=dd== And it is being offered
===== -------------------- 1 line(s) not displayed --------------------
=dd== For more information, please call.
```

When SCOPE DISPLAY is set, the shadow line is not affected by the execution of the block entry. Please note that the shadow line remains in the file in this instance.

```
===== The resort is comprised of 500 acres.
===== It has a private pond, waterfalls, brooks, and woodlands.
===== It is just one step away from hiking, swimming, and skating.
===== -------------------- 1 line(s) not displayed --------------------
```

However, when the same block entry is executed while SCOPE ALL is set, the shadow line is affected by the execution of the block entry.

```
===== The resort is comprised of 500 acres.
===== It has a private pond, waterfalls, brooks, and woodlands.
===== It is just one step away from hiking, swimming, and skating.
===== * * * End of File * * *
```

Appendix C. CMS Editor (EDIT) Migration Mode

To edit a file in EDIT migration mode, issue the CMS command, EDIT, the same way that you would when you normally invoke the CMS editor. The XEDIT editor automatically places you in EDIT migration mode, in which you can issue all of the EDIT subcommands; you can also issue any XEDIT subcommand that does not have the same name as an EDIT subcommand.

In addition, EDIT migration mode provides full screen editing capabilities, that is, you can type over data on any file line that is displayed on the screen.

If you want to invoke the old CMS editor (instead of XEDIT in EDIT migration mode) for a particular file, you can specify "OLD" as an option on the EDIT command. For example:

```
EDIT fn ft (OLD
```

The old CMS editor has not been enhanced for VM/SP and will not be enhanced in future releases of VM/SP. Specifically, the CMS editor will not include support for new display devices.

Usage Notes

1. The following EDIT subcommands are executed in EDIT migration mode the same way they are executed under the CMS editor:

```
ALTER       FNAME       PROMPT      STACK
AUTOSAVE    FORMAT      QUIT        TABSET
BACKWARD    FORWARD     RECFM       TOP
BOTTOM      GETFILE     RENUM       TRUNC
CASE        IMAGE       REPEAT      TYPE
CHANGE      INPUT       REPLACE     UP
CMS         LINEMODE    RESTORE     X,Y
DELETE      LOCATE      RETURN      ZONE
DOWN        LONG        REUSE       ?
DSTRING     NEXT        SAVE        $XXXX *
FIND        OVERLAY     SERIAL
FMODE       PRESERVE    SHORT
```

   * When the editor parses input that begins with "$", it interprets the input as an EXEC and ignores any IMPCMSCP or MACRO settings.

2. The following EDIT subcommands are executed slightly differently in EDIT migration mode:

   SCROLL/SCROLLUP
   Under the CMS editor, these subcommands scroll the file one full screen. In EDIT migration mode, they scroll the screen one full screen minus one line, so that the last (or first) line on the previous screen appears on the new display.

   VERIFY
   Under the CMS editor, the operands ON and OFF are ignored if the VERIFY subcommand is issued from a display terminal. In EDIT migration mode, ON and OFF are handled the same way on a display as they are on a typewriter terminal.

3. Any XEDIT subcommand that does not appear in the list above can be issued in EDIT migration mode.

4. In EDIT migration mode, you can issue the XEDIT subcommand HELP to request information on EDIT subcommands only. You cannot request a HELP display for XEDIT subcommands.
5. In the EDIT command, the default file mode is "*" instead of "A1."
6. The screen differs from the CMS editor's screen in the following ways:
   a. The file identification line (the first line of the screen) contains the following additional information: the truncation column (Trunc = nn); the current number of lines in the file (Size = nn); the file line number of the current line (Line = nn); the column number of the current column (Col = nn), and the alteration count (Alt = nn).
   b. The command line contains an arrow (= = = = >).
   c. The lower right hand corner displays the status of the editing session, for example, input mode or edit mode.
7. When you issue EDIT, the linend character default is "ON". When editing files which contain characters the editor recognizes as linend characters, you may want to SET LINEND OFF or change it to another character. See SET LINEND subcommand in this manual. In line mode, a CP TERMINAL LINEND OFF must also be issued to cancel the effect of the default linend character. See the TERMINAL command in the *VM/SP CP General User Command Reference.*

Notes for Macro Writers

When an EDIT command is issued from an EXEC file, the CMS editor is invoked. To invoke EDIT migration mode, you must issue the following statement in your EXEC file:

```
EXEC EDIT ...
```

Appendix D. Migrating from EDIT to XEDIT

Table 2 lists the EDIT subcommands and their XEDIT counterparts.

Many of the subcommand names are the same; however, the operands are usually different. Refer to the subcommand descriptions in this book for complete information on using the XEDIT subcommands.

Table 2 (Page 1 of 2). EDIT Migration Chart

If you used this EDIT command:	**Now use this XEDIT command:**
ALTER	ALTER
AUTOSAVE	SET AUTOSAVE
BACKWARD	UP
BOTTOM	BOTTOM
CASE	SET CASE
CHANGE	CHANGE (G not supported)
CMS	CMS
DELETE	DELETE
DOWN	DOWN
DSTRING	DELETE
FILE	FILE
FMODE	SET FMODE
FNAME	SET FNAME
FORMAT	SET TERMINAL
FORWARD	DOWN
GETFILE	GET
IMAGE	SET IMAGE
INPUT	INPUT
LINEMODE	Not supported
LOCATE	LOCATE
LONG	SET MSGMODE ON LONG
NEXT	NEXT
OVERLAY	OVERLAY
PRESERVE	PRESERVE
PROMPT	Not supported
QUIT	QUIT
RECFM	SET RECFM
RENUM	RENUM
REPEAT	REPEAT (repeat any previous subcommands)
REPLACE	REPLACE

Table 2 (Page 2 of 2). EDIT Migration Chart

If you used this EDIT command:	**Now use this XEDIT command:**
RESTORE	RESTORE
RETURN	RETURN
REUSE (=)	=
SAVE	SAVE
SCROLL	FORWARD
SCROLLUP	BACKWARD
SERIAL	SET SERIAL
SHORT	SET MSGMODE ON SHORT
STACK	STACK (STACK string not supported)
TABSET	SET TABS
TOP	TOP
TRUNC	SET TRUNC
TYPE	TYPE (second operand not supported)
UP	UP
VERIFY	SET VERIFY
X or Y	Can be done via SET SYNONYM and/or REPEAT
ZONE	SET ZONE
?	?
nnnn	:nnnn (nnnn is equivalent to +nnnn)
$DUP	DUPLICATE
$MOVE	MOVE

Appendix E. Optimizing Macros

The XEDIT macro VMFOPT can be used to improve the performance of XEDIT macros. Conversely, a macro optimized by the VMFOPT macro can be restored to its original form by executing the VMFDEOPT macro.

The following XEDIT macros have already been optimized:

```
CMSEDIT
VMFOPT
VMFDEOPT
```

These macros contain the following statements, which are inserted during the optimizing process; they indicate that if a macro needs to be changed, it must first be deoptimized (by using VMFDEOPT) and then reoptimized (by using VMFOPT).

```
*%OPTIMIZED AT 14:13:12 ON 80/01/29
*%       N O T I C E:
*% THIS MACRO HAS BEEN OPTIMIZED USING THE XEDIT MACRO - VMFOPT
*% DE-OPTIMIZE THIS MACRO BEFORE MAKING ANY CHANGES USING - VMFDEOPT
```

Note: VMFOPT and VMFDEOPT will execute satisfactorily only on the macros listed above.

The VMFOPT Macro

Format

VMFOPT	

The VMFOPT macro improves performance in the following ways:

- It replaces labels with line numbers in EXEC 2 &GOTO statements. In other words, an EXEC 2 statement "&GOTO −LABEL" is replaced by an equivalent statement, "&GOTO linenumber −LABEL". For example, the following EXEC 2 statement:

  ```
  &GOTO -EXIT
  ```

  is replaced by:

  ```
  &GOTO 182 -EXIT
  ```

  where the label "−EXIT" is on line 182. Notice that the label name (EXIT) is kept, so that the macro can be deoptimized, if necessary.

- It recomputes existing "&GOTO linenumber" statements whose targets may have shifted because of extra lines inserted by VMFOPT.

- It is also able to optimize label variables of the form "&GOTO −&X". Targets for label variables can be declared as label constants, using the optimizer control command, *%LABELS. This command causes VMFOPT to insert EXEC 2 statements that set up special variables. The name of these special variables contains the unprintable character X'E0' to avoid confusion with user-defined variables.

For example:

```
Prior to optimizing:

*%LABELS -BOTTOM -CANCEL -CASE -CHANGE -CLINE -CMD

When optimized:

*%LABELS -BOTTOM -CANCEL -CASE -CHANGE -CLINE -CMD  SYN -CMS
 &STACK LIFO  187 194 199 221 265 287
 &READ VARS  & -BOTTOM & -CANCEL & -CASE & -CHANGE & -CLINE & -CMD
```

The VMFDEOPT Macro

Macros optimized by VMFOPT can be restored to their original form by executing the VMFDEOPT macro.

Format

VMFDEOPT	

After the VMFDEOPT macro is executed, all &GOTO statements are restored to their original form, and any extra lines added by VMFOPT (for example, the optimization statement) are removed.

When you wish to change an optimized macro, first deoptimize it using VMFDEOPT and then, after the changes are made, reoptimize it using VMFOPT.

Appendix F. Using Double-Byte Character Sets

Many languages have more characters than can be displayed using one-byte codes (KANJI, for example). A Double-Byte Character Set (DBCS) is used to represent these languages. These characters can be displayed on terminals that support Double-Byte Character Sets, such as the IBM 5550 Multistation. Each double-byte character occupies two columns on the screen. DBCS characters and characters from languages with one-byte codes can be mixed within a string.

Specifying DBCS Characters in a File

In XEDIT, to distinguish DBCS characters from one-byte EBCDIC characters, DBCS strings are enclosed with a shift-out (SO) character and a shift-in (SI) character.

Character	Hexadecimal	Description
▮	X'0E'	shift–out (SO)
▮	X'0F'	shift–in (SI)

Key to Conventions Used in This Section

Character	Represents a
e	Lowercase EBCDIC character
E	Uppercase EBCDIC character
Ｅ	One DBCS character (occupies two columns)
ＡＢＣ	A DBCS string

Editing DBCS strings

In order for you to see DBCS strings and for XEDIT to recognize DBCS strings in a file, the ETMODE setting must be ON.

The initial setting is based on whether the terminal can display double-byte characters. If it can, the initial setting is ON; if not, the setting is OFF.

In order for XEDIT to properly display and manipulate DBCS strings, when ETMODE is on, it does the following:

- Drops contiguous shift-out and shift-in characters after executing a subcommand (with a few exceptions)
- Generates proper pairings of shift-out and shift-in characters after executing a subcommand
- Provides special consideration for shift-out and shift-in when doing a target search.

Dropping Contiguous Shift-out and Shift-in Characters

When you issue a subcommand, XEDIT checks for contiguous shift-out and shift-in characters generated by the operation. Contiguous shift-out and shift-in characters are deleted and the remainder of the line shifts to the left two bytes, (except with the CREPLACE subcommand, which changes these characters to blanks). For example, if you were to join lines 10 and 11 at column 12:

```
      |...+....1....+....2....+....3....+....4....
00010 eee▮ＡＡＡ▮
00011 ▮ＢＢＢ▮eee
```

The result would be:

```
      |...+....1....+....2....+....3....+....4....
00010 eee▮ＡＡＡＢＢＢ▮eee
```

The contiguous shift-out and shift-in characters were removed.

Pairing Shift-out and Shift-in Characters

XEDIT properly pairs shift-out and shift-in characters after executing a subcommand. If a subcommand results in the deletion of a shift-out or shift-in character, then the appropriate character is inserted so that a DBCS string is always between shift-out and shift-in characters. For example, if you were to split lines 10 and 11 at column 11:

```
      |...+....1....+....2....+....3....+....4....
00010 eee▮ＡＡＡＢＢＢ▮eee
```

The result would be:

```
      |...+....1....+....2....+....3....+....4....
00010 eee▮ＡＡＡ▮
00011 ▮ＢＢＢ▮eee
```

Target Searches with Shift-out and Shift-in Characters

When XEDIT scans a file for a string target, an SO or SI is ignored in the following cases:

- The first character of a string target is an SO.
- The last character of a string target is an SI.
- The SO or SI is specified immediately preceding or immediately following:
  - An arbitrary character when ARBCHAR is ON
  - An extended arbitrary character when ETMODE and ETARBCH are ON.

For example:

`LOCATE /▮Ａ▮/` will locate ▮ＢＡＢ▮.

The shift-out and shift-in characters of the target are ignored during the search.

Using XEDIT Subcommands with DBCS

The following subcommands can be used in extended mode (ETMODE ON).

Add	Query	SET PENDing
ALL	QUIT	SET PFn
ALter	READ	SET Point
BAckward	RECover	SET PREfix
Bottom	REFRESH	SET RANge
CANCEL	RENum	SET RECFm
CAppend	REPEat	SET REMOte
CDelete	Replace	SET RESERved
CFirst	RESet	SET SCALe
Change	RESTore	SET SCOPE
CInsert	RGTLEFT	SET SCReen
CLAst	RIght	SET SELect
CLocate	SAVE	SET SERial
CMS	SCHANGE	SET SHADow
CMSG	SET ALT	SET SIDcode
COMMAND	SET APL	SET SPAN
COpy	SET ARBchar	SET SPILL
COUnt	SET AUtosave	SET STAY
CP	SET BRKkey	SET STReam
CReplace	SET CASE	SET SYNonym
CURsor	SET CMDline	SET TABLine
DELete	SET COLOR	SET TABS
Down	SET COLPtr	SET TERMinal
DUPlicat	SET CTLchar	SET TEXT
EMSG	SET CURLine	SET TOFEOF
EXTract	SET DISPlay	SET TRANSLat
FILE	SET ENTer	SET TRunc
Find	SET ESCape	SET VARblank
FINDUp	SET ETARBCH	SET Verify
FOrward	SET ETMODE	SET WRap
GET	SET FILler	SET Zone
Help	SET FMode	SET =
Input	SET FName	SHift
Join	SET FType	SI
LEft	SET FULLread	SOS
LOAD	SET HEX	SPlit
Locate	SET IMage	SPLTJOIN
LOWercas	SET IMPcmscp	STAck
LPrefix	SET LASTLorc	STATus
MACRO	SET LINENd	TOP
MODify	SET LRecl	TRAnsfer
MOve	SET MACRO	Type
MSG	SET MASK	Up
Next	SET MSGLine	UPPercas
NFind	SET MSGMode	Xedit

NFINDUp	SET NONDisp	&
PARSE	SET NULls	=
PREServe	SET NUMber	?
PURge	SET PAn	.XXXX
PUT, PUTD	SET PACK	

Subcommands that Support SET ETMODE

The following subcommands will return the ETMODE setting.

```
EXTRACT
MODIFY
QUERY
STATUS
```

Subcommand	Results
EXTRACT /ETMODE/	returns ON or OFF as specified by the SET ETMODE subcommand. ETMODE.0 number of variables returned .1 ON|OFF
MODIFY ETMODE	displays SET ETMODE ON or SET ETMODE OFF on the command line.
QUERY ETMODE	displays ON or OFF as defined by the SET ETMODE subcommand.
STATUS	returns the setting of ETMODE as well as the settings of the other SET subcommand options.

Subcommands that Support SET ETARBCH

The following subcommands will recognize the ETARBCH setting.

```
EXTRACT
MODIFY
PRESERVE/RESTORE
QUERY
```

Subcommand	Results
EXTRACT /ETARBCH/	returns "ON" or "OFF" and the extended arbitrary character specified in the SET ETARBCH subcommand. ETARBCH.0 number of variables returned .1 ON|OFF .2 extended arbitrary character enclosed by a shift-in and a shift-out character
MODIFY ETARBCH	returns SET ETARBCH ON|OFF character on the command line.
PRESERVE	saves the ETARBCH setting until a subsequent RESTORE subcommand is issued.

RESTORE	restores the ETARBCH setting to the value it had when the PRESERVE subcommand was issued.
QUERY ETARBCH	displays ON or OFF and the extended arbitrary character defined by the SET ETARBCH subcommand.

XEDIT subcommands with SET ETMODE ON

The following subcommands are described in this section.

CAppend	Input	SET SIDcode
CDelete	Join	SET SPILL
CFirst	LEft	SET SYNonym
Change	Locate	SET TRunc
CInsert	LOWercas	SET Verify
CLAst	NFind, NFINDUp	SET Zone
CLocate	PUT, PUTD	SHift
CReplace	Replace	SPlit
CURsor	SET LRecl	SPLTJOIN
Find, FINDUp	SET SERial	UPPercas
GET		

CAPPEND

When you append a DBCS string to another DBCS string, the shift-in and shift-out characters between the two strings are dropped. For example, if you issue

```
CAPPEND ▮ＢＢ▮
```

against the current line:

```
      |...+....1....+....2....+....3....+....4....
00010 eee▮ＡＡＡＡＡ▮
```

The result will be:

```
      <...+....1....|....2....+....3....+....4....
00010 eee▮ＡＡＡＡＡＢＢ▮
```

CDELETE

When you specify a column target with the CDELETE subcommand, that column is checked to see if it is the second byte of a DBCS character. If so, the previous column is considered the target. CDELETE will not delete a shift-out or shift-in character if it is required to maintain the integrity of the DBCS string.

For example, if the column pointer is at column 1 and you issue the subcommand:

```
CDELETE :8
```

against the current line:

```
      |...+....1....+....2....+....3....+....4....
00010 eee▮ＡＢＣＤＥ▮
```

The result will be:

```
      |...+....1....+....2....+....3....+....4....
00010 ▮ＢＣＤＥ▮
```

CFIRST

If you issue CFIRST and the left zone is the second byte of a DBCS character, then the column pointer will be repositioned at the left zone plus one.

CHANGE

When you issue the CHANGE subcommand, DBCS strings can be in the string to be changed and in the new string.

Example:

Subcommand		From	To
C/▮Ａ▮/▮Ｂ▮/	will change	▮Ａ▮	▮Ｂ▮
		▮ＡＣＣ▮	▮ＢＣＣ▮
		▮ＣＣＡ▮	▮ＣＣＢ▮
		▮ＣＡＣ▮	▮ＣＢＣ▮
C/▮Ａ▮e/▮Ｂ▮E/	will change	▮Ａ▮eeee	▮Ｂ▮Eeee
		▮ＣＡ▮ee	▮ＣＢ▮Ee
	will not change	▮ＣＡＣ▮ee	▮ＣＢＣ▮ Ee

You can specify an extended arbitrary character (with SET ETARBCH) in a DBCS string in the same fashion as the arbitrary character (SET ARBCHAR) is used in EBCDIC strings to specify that any characters may appear in the matching string in the file. Following are examples of using CHANGE with arbitrary and extended arbitrary characters.

Example:

Subcommand		From	To
C/e$e/▮Ａ￥Ｃ▮/	will change	e▮Ｂ▮e	▮ＡＢＣ▮
C/e▮￥▮e/▮Ａ▮$▮Ｃ▮/	will change	eEe	▮Ａ▮E▮Ｃ▮
C/e$e/▮Ａ▮$▮Ｃ▮/	will change	eE▮Ｂ▮Ee	▮Ａ▮E▮Ｂ▮E▮Ｃ▮
C/e▮￥▮e/▮Ａ￥Ｃ▮/	will change	e▮Ｂ▮E▮Ｂ▮e	▮ＡＢ▮E▮ＢＣ▮
C/▮Ａ￥Ｃ▮/e$e/	will change	▮Ａ▮E▮Ｃ▮	eEe
C/▮Ａ▮$▮Ｃ▮/e▮￥▮e/	will change	▮ＡＢＣ▮	e▮Ｂ▮e
C/▮Ａ▮$▮Ｃ▮/e$e/	will change	▮ＡＢ▮E▮ＢＣ▮	e▮Ｂ▮E▮Ｂ▮e
C/▮Ａ￥Ｃ▮/e▮￥▮e/	will change	▮Ａ▮E▮Ｂ▮E▮Ｃ▮	eE▮Ｂ▮Ee

When a double-byte character is at or beyond the zone boundary, then it will not be changed.

Example:

Subcommand		From	To
C/▮Ｂ▮/▮Ｃ▮/	will not change	`...|<....>....+.` ▮ＡＢＣＤＥ▮	`...|<....>....+.` ▮ＡＣＣＤＥ▮
C/▮Ｅ▮/▮Ｆ▮/	will not change	`...|<....>....+.` ▮ＡＢＣＤＥ▮	`...|<....>....+.` ▮ＡＢＣＤＦ▮
C/▮ＣＤ▮/▮ＡＡ▮/	will change	`...|<....>....+.` ▮ＡＢＣＤＥ▮	`...|<....>....+.` ▮ＡＢＡＡＥ▮

CINSERT

When a DBCS string is inserted into another DBCS string, the shift-out and shift-in characters are not inserted. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２３４５６７▮eee
```

Issuing the subcommand:

```
CINSERT ▮ＡＡ▮
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２３ＡＡ４５６７▮eee
```

When an EBCDIC string is inserted into a DBCS string, appropriate shift-in and shift-out characters are inserted to separate the original DBCS string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２３４５６７▮eee
```

Issuing the subcommand:

```
CINSERT eeee
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２３▮eeee▮４５６７▮eee
```

When an DBCS string is inserted into an EBCDIC string, the shift-out and shift-in characters are inserted as part of the string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eeeeeeeeeeeeeeeeeeeeee
```

Issuing the subcommand:

```
CINSERT ▮ＡＡ▮
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eeeeeeeeee▮ＡＡ▮eeeeeeeeeeee
```

CLAST

When you use the CLAST subcommand to move the column pointer to the end of the zone, a check is made to see if the right zone is at the second byte of a DBCS character. If the right zone is the second byte of a DBCS character, the column pointer is adjusted to the first byte of the DBCS character (at zone right less one).

CLOCATE

If the column target is set on the second byte of a DBCS character, the column pointer is repositioned to the previous column. If the column pointer is at TOL or EOL, no adjustment occurs.

CREPLACE

When you replace part of a DBCS string with EBCDIC characters, shift-out and shift-in characters are generated to maintain proper pairing of the characters. If the second byte of a DBCS character remains after the subcommand executes, it is replaced by a blank. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７␏eee
```

Issuing the subcommand:

```
CREPLACE eee
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３␏eee ␎７␏eee
```

When you replace one DBCS string with another DBCS string the shift-out and shift-in characters are not used as a part of the replacement string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７␏eee
```

Issuing the subcommand:

```
CREPLACE ␎ＡＡ␏
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３ＡＡ６７␏eee
```

However, when a DBCS string replaces part of an EBCDIC string, the shift-out and shift-in characters are included in the replacement string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２␏eeeeeeee␎３４␏eee
```

Issuing the subcommand:

```
CREPLACE ▮Ａ▮
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２▮e▮Ａ▮ee▮３４▮eee
```

CURSOR

The CURSOR subcommand will adjust the cursor position so that the character boundaries of DBCS characters are respected. For example (the cursor is at column 1):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２３４５６７８９▮eee
```

Issuing the subcommand:

```
CURSOR FILE 10 20
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▮１２３４５６７８９▮eee
```

EXTRACT

A shift-out character, as the self-defining delimiter in the EXTRACT subcommand, results in a return code of 5. EXTRACT.0 will be set to 1 and EXTRACT.1 will be set to the invalid delimiter (the shift-out character).

FIND, FINDUP

The shift-in character is ignored during a search for a DBCS string. Issuing:

```
FIND ▮１２▮
```

would locate either:

```
▮１２▮ or ▮１２３▮
```

GET

If truncation of a DBCS string occurs when inserting lines from another file, a shift-in character is inserted to close the DBCS string.

INPUT

If truncation of a DBCS string occurs when inserting a line, a shift-in character is inserted to close the DBCS string.

JOIN

When two DBCS strings are joined, the shift-in character from the first string and the shift-out character from the second string are dropped. For example (the column pointer is at column 24):

```
      <...+....1....+....2...|+....3....+....4....
00010 eee■１２３４５６７８９■eee
00011 ■ＡＡＡＡＡＡＡ■
```

Issuing the subcommand:

```
JOIN COLUMN
```

results in:

```
      <...+....1....+....2..|.+....3....+....4....
00010 eee■１２３４５６７８９ＡＡＡＡＡＡＡ■
```

When an EBCDIC string is joined to a DBCS string, a shift-in character is inserted at the end of the DBCS string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee■１２３４５６７８９■eee
00011 EEEEEEEEEEEEE
```

Issuing the subcommand:

```
JOIN COLUMN
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee■１２３■EEEEEEEEEEEEE
```

LEFT

If issuing a LEFT subcommand results in the display of “negative columns”, you should not type data in these columns or you may lose DBCS strings.

LOCATE

The ability to locate a line via a target is one of the editor's most useful functions. A target is not only used as the operand of the LOCATE subcommand, but also as an operand in many other XEDIT subcommands. The following XEDIT subcommands will accept DBCS strings in targets.

Subcommands that Accept DBCS Strings as Targets:

ALL	DELete	REPEat
ALter	DUPlicat	SET RANge
CDelete	EXTract	SET SELect
Change	Locate	SHift
CLocate	LOWercas	STAck
COpy	MOve	Type
COUnt	PUT, PUTD	UPPercas.

Special Considerations for Targets: When XEDIT scans a file for a string target, an SO or SI is ignored in the following cases:

- The first character of a string target is an SO.
- The last character of a string target is an SI.
- The SO or SI is specified immediately preceding or immediately following:
  - An EBCDIC or double-byte blank when VARBLANK is ON
  - An arbitrary character when ARBCHAR is ON
  - An extended arbitrary character when ETMODE and ETARBCH are ON.

  SET CASE

  The IGNORE and RESPECT settings of the SET CASE subcommand have no effect when searching for a DBCS string, because there are no lowercase/uppercase pairs for double-byte characters. For example:

  ```
  SET CASE M IGNORE
  /e■ａ■e/
  ```

  will locate:

  ```
  E■ａ■E
  ```

  It will not locate:

  ```
  E■Ａ■E
  ```

SET VARBLANK

You can use the SET VARBLANK ON or OFF subcommand to control whether the number of blank characters is significant when searching for a string. For example:

```
SET VARBLANK ON
/e ▮Ａ▮/
```

would locate in the text either of the following:

```
e ▮  ▮ ▮Ａ▮
e▮  ▮ ▮  Ａ▮
```

SET SPAN

DBCS strings at the end of a line and at the beginning of the next line are treated as one string when SET SPAN ON is specified. The shift-in character at the end of the first line and the shift-out character at the beginning of the second line are ignored. If SET SPAN ON BLANK is specified, then one double-byte blank is inserted between the first and the second string.

Example:

Command	Target	Lines in a File	Match / No Match
SET SPAN ON NOBLANK 2	/▮ＡＢ▮/	00010 ▮１Ａ▮ 00011 ▮Ｂ１▮	Match
	/▮Ａ　Ｂ▮/	00010 ▮１Ａ▮ 00011 ▮Ｂ１▮	No Match
SET SPAN ON BLANK 2	/▮ＡＢ▮/	00010 ▮１Ａ▮ 00011 ▮Ｂ１▮	No Match
	/▮Ａ　Ｂ▮/	00010 ▮１Ａ▮ 00011 ▮Ｂ１▮	Match

SET ETARBCH

You can specify an extended arbitrary character (with SET ETARBCH) in a DBCS string in the same fashion as the arbitrary character (SET ARBCHAR) is used in EBCDIC strings to specify that any characters may appear in the matching string in the file. For example:

```
SET ETARBCH ON
/▮Ａ￥Ａ▮/
```

would locate in the text any of the following:

```
▮ＯＡＯＡＯ▮
▮Ａ▮eee▮Ａ▮
▮Ａ▮e▮Ｏ▮e▮Ａ▮
```

SET ZONE

When a double-byte character is at or beyond the zone boundary, then it will not be located. For example:

```
/▮２▮/      will not locate      ...|<....>....+....
                                 ▮１２３４５６▮

/▮５▮/      will not locate      ...|<....>....+....
                                 ▮１２３４５６▮

/▮２▮/      will locate          ..|<+....1>...+....
                                 ▮１２３４５６▮

/▮５▮/      will locate          ..|<+....1>...+....
                                 ▮１２３４５６▮
```

LOWERCAS

When you issue the LOWERCAS subcommand, DBCS strings are excluded from translation. Double-byte characters have no lowercase or uppercase pairs.

NFIND, NFINDUP

The shift-in character is ignored during a search for a DBCS string. Issuing:

```
FIND ▮ＡＢ▮
```

would locate either:

```
▮ＡＢ▮   or   ▮ＡＢＣ▮
```

PUT, PUTD

If you use PUT or PUTD to append data to a fixed format file, DBCS strings may be truncated. You will receive a message warning you that DBCS strings may have been lost due to truncation.

REPLACE

If a DBCS string is truncated when a line is inserted, a shift-in character is inserted to close the DBCS string.

SET LRECL

If you use SET LRECL to shorten the logical record length of a file being edited, you will receive a message warning you that DBCS strings may be lost when the file is saved or filed.

SET SERIAL

If you specify a serial string that is longer than eight characters, only the first eight characters are recognized by XEDIT. When ETMODE is ON and the eighth character of the serial string falls within a DBCS string, the editor adjusts the serial string to maintain the integrity of the DBCS string. For example, if you specify the serial string:

```
ab▮１２３▮
```

XEDIT will recognize it as the following:

```
ab▮１２▮
```

SET SIDCODE

If you specify a sidcode string that is longer than eight characters, only the first eight characters are recognized by XEDIT. When ETMODE is ON and the eighth character of the sidcode string falls within a DBCS string, the editor adjusts the sidcode string to maintain the integrity of the DBCS string. For example, if you specify the sidcode string:

```
ab▮１２３▮
```

XEDIT will recognize it as the following:

```
ab▮１２▮
```

SET SPILL

When spilling occurs within a DBCS string, a shift-in character is inserted at the end of the first line and a shift-out character is inserted at the front of the remaining string that is spilled.

SET SYNONYM

When ETMODE is ON, a synonym name may consist of single-byte characters, double-byte characters, or a combination of single-byte and double-byte characters. However, an abbreviation of a synonym name that includes a DBCS string, must use the complete DBCS string contained in the synonym. For example, the following synonym:

```
a▮１２▮b
```

can be abbreviated as:

`a` or `ab▮１２▮`

but cannot be abbreviated as:

```
a▮１▮
```

If you specify a synonym name that is longer than eight characters, only the first eight characters are recognized by XEDIT. When ETMODE is ON and the eighth character of the synonym name falls within a DBCS string, the editor adjusts the synonym name to maintain the integrity of the DBCS string. For example, if you specify the synonym:

```
ab▮１２３▮
```

XEDIT will recognize it as the following:

```
ab▮１２▮
```

SET TRUNC

Certain XEDIT subcommands insert characters in a file line shifting existing data to the right. If a DBCS string is shifted beyond the truncation column (defined by SET TRUNC), a shift-in character is generated to maintain the proper pairing of the characters. For example, issuing the subcommands:

```
SET TRUNC 15
CLOCATE :4
```

on the following string:

```
      <..|+....1....>....2....+....3....+....4....
00010 eee▮ＡＢＣＤ▮
```

Then, issuing the subcommand:

```
CINSERT eeeee
```

results in:

```
      <..|+....1....>....2....+....3....+....4....
00010 eeeeeeee▮ＡＢ▮
```

The last DBCS character prior to the truncation column is replaced by a shift-in character and a blank.

Other XEDIT subcommands shorten the length of the file line and insert blanks at the truncation column. If the truncation column falls within a DBCS string, then blanks are inserted after the shift-in that follows the truncation column. The CDELETE, CREPLACE, CHANGE, and SHIFT subcommands treat the TRUNC setting in this manner.

For example, issuing the subcommands:

```
ZONE 11 *
SET TRUNC 20
CLOCATE :17
```

so that the truncation column falls within a DBCS string:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eeeeeeeeeeeee■ＡＢＣＤ■eeeeeeeeeee
```

Then, issuing the subcommand:

```
CDELETE 2
```

results in:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eeeeeeeeeeeee■ＡＣＤ■  eeeeeeeeeee
```

The line shifted starting at the shift-in character instead of the truncation column (column 20 in this example).

SET VERIFY

When ETMODE is OFF, you can issue SET VERIFY to display multiple column pairs. Then, if you issue SET ETMODE ON, only the first pair of verify columns continues to be displayed. However, when ETMODE is ON and you issue SET VERIFY with multiple column pairs, you will get an error message.

For example, when ETMODE is OFF and if you issue the subcommand:

```
V 1 20 40 50
```

columns 1 through 20 and 40 through 50 are displayed.

Then, if you issue:

```
SET ETMODE ON
```

only columns 1 through 20 continue to be displayed.

When ETMODE is ON and the column(s) specified by SET VERIFY falls within a DBCS string, overtyping to change DBCS characters to EBCDIC characters on the first or last column may change the original character that does not appear on the screen. For example:

```
00010 eeeeeeeeeeeee■ＡＢＣＤ■eeeeeeeeeee
      ....+....1....+....2....+....3....+....4....
```

issuing the following:

```
SET VERIFY 15
```

results in:

```
00010  ▮ＢＣＤ▮eeeeeeeeeee
       +....2....+....3....+....4....
```

If you overtype the line as follows:

```
00010  a ▮ＣＤ▮eeeeeeeeeee
       +....2....+....3....+....4....
```

then you will have changed the first two double-byte characters:

```
00010 eeeeeeeeeeeeee▮▮a ▮ＣＤ▮eeeeeeeeeee
      |...+....1....+....2....+....3....+....4....
```

SET ZONE

Certain XEDIT subcommands ignore data outside the left and/or right zone (defined by SET ZONE). If the zone column falls within a DBCS string, then the subcommand executes in the entire DBCS string up to and including the shift-in or shift-out character. The CDELETE and SHIFT subcommands treat the ZONE settings in this manner. If the left zone falls within a DBCS string, then the shift-out character is treated as the left zone. If the right zone falls within a DBCS string, then the shift-in character is treated as the right zone. For example, issuing the subcommands:

```
SET ZONE 11 20
CLOCATE :17
```

so that a zone column falls within a DBCS string:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eee▮ＡＢＣＤ▮eeeeeeeeeee
```

Then, issuing the subcommand:

```
SHIFT RIGHT 1
```

results in:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eee ▮ＡＢＣＤ▮eeeeeeeeeee
```

The line shifted starting at the shift-out character instead of the left zone (column 11 in this example).

SHIFT

When shifting data beyond the file boundary (left of zone left or right of the truncation column), shift-out and shift-in characters are generated to maintain proper pairing of the characters. For example (the column pointer is at column 4):

```
      <..|+....1....+....2....+....3....+....4....
00010 eee▮１２３４５６７８９▮eee
```

Then issuing the subcommand:

```
SHIFT LEFT 5
```

results in:

```
      <..|+....1....+....2....+....3....+....4....
00010 ▮１２３４５６７８９▮eee
```

For another example (column pointer is at column 4, truncation column is at column 25):

```
      <..|+....1....+....2....>....3....+....4....
00010 eee▮１２３４５６７８９▮ee
```

Then issuing the subcommand:

```
SHIFT RIGHT 5
```

results in:

```
      <..|+....1....+....2....>....3....+....4....
00010      eee▮１２３４５６７▮
```

In this example the last DBCS character prior to the truncation column is replaced by a shift-in character and a blank.

SPLIT

When you split a DBCS string, a shift-in character is inserted at the end of the original line and a shift-out character is inserted at the first column of the new line. For example (column pointer is at column 11):

```
      <...+....1|...+....2....>....3....+....4....
00010 eee▮ＡＢＣＤＥＦＧＨＩ▮ee
```

and you issue the subcommand:

```
SPLIT COLUMN
```

the result is:

```
00010 eee■ＡＢＣ■
      <...+....1|...+....2....>....3....+....4....
00011 ■ＤＥＦＧＨＩ■ee
```

When the split places all the DBCS characters on a new line the shift-out character is deleted from the original line and inserted at the beginning of the new line. For example (column pointer is at column 5):

```
      <...|....1....+....2....>....3....+....4....
00010 eee■ＡＢＣＤＥＦＧＨＩ■ee
```

When you issue the subcommand:

```
SPLIT COLUMN
```

the result is:

```
00010 eee
      <...|....1....+....2....>....3....+....4....
00011 ■ＡＢＣＤＥＦＧＨＩ■ee
```

SPLTJOIN

When you split a DBCS string, a shift-in character is inserted at the end of the original line and a shift-out character is inserted at the first column of the new line.

When two DBCS strings are joined, the shift-in character from the first string and the shift-out character from the second string is dropped. When an EBCDIC string is joined to a DBCS string, a shift-in character is inserted at the end of the DBCS string. For examples of splitting and joining lines, refer to the descriptions of SPLIT and JOIN in this section.

UPPERCAS

When you issue the UPPERCAS subcommand, DBCS strings are excluded from translation. Double-byte characters have no lowercase or uppercase pairs.

Error Message 580E

When ETMODE is ON the following error message may be issued from any XEDIT subcommand or macro:

```
580E Invalid string: [Shift-out (SO) is not a valid
     delimiter | Unmatched shift-out (SO) and shift-in
     (SI) | Odd number of characters between SO and
     SI | Invalid double-byte character(s)].,RC=5
```

Restrictions for DBCS

1. On terminals without DBCS character support, such as a 3277, DBCS data will not be displayed correctly.
2. The XEDIT sort algorithm does not apply to DBCS codes.
3. A shift-out character (X'0E') may not be used as a self-defining delimiter. For example:

   ```
   LOCATE ▮Ａ▮
   ```

   is not valid. You must specify a delimiter such as a slash (/). For example:

   ```
   LOCATE /▮Ａ▮/
   ```

   is valid.
4. DBCS strings cannot be specified as a file name, file type, or a file mode because of restrictions in the CMS file system.
5. XEDIT does not allow DBCS strings to span lines in a file. The DBCS string must be contained in a single line so that shift-out and shift-in pairs are maintained on each line in the file. Consider this when editing a PL/I source statement.
6. Data stacked by the STACK, READ, and TRANSFER subcommands may be truncated because of the CMS stack limitation of 255 characters. When you write macros, keep in mind that the data being stacked will be truncated if it exceeds 255 characters.
7. The following subcommands are not allowed in extended mode (SET ETMODE ON). You are able to execute these subcommands with SET ETMODE OFF. However, DBCS strings are treated as EBCDIC data and you may get unpredictable results with respect to DBCS data.

   ```
   COMPress          HEXType           POWerinp
   COVerlay          MErge             SORT
   EXPand            Overlay
   ```
8. When SET ETMODE is on, XEDIT does not scan for control characters (defined by SET CTLCHAR) within a DBCS string.
9. When a DBCS string is specified in a PARSE subcommand, XEDIT stops searching for the self-defining delimiter within a DBCS string.

DBCS Strings

Before XEDIT displays any data or accepts any operands, it verifies that the DBCS string is valid. A valid DBCS string has the following characteristics.

1. All DBCS strings <u>must</u> be preceded by a shift-out character and followed by a shift-in character.
2. The length of the string must be even.

3. The hex code for a DBCS character must be within the following range:

   ```
   first  byte    X'41' - X'FE'
   second byte    X'41' - X'FE'
                  X'4040'          (DBCS blank)
                  X'0000'          (DBCS null)
   ```

If any of the above conditions are not met when using DBCS strings with an XEDIT subcommand, then an error message is issued.

When you are displaying data, if any invalid codes are found, then the DBCS character is replaced with a DBCS NONDISP character (X'427F'). Invalid DBCS strings will be displayed as EBCDIC data. The shift-out and shift-in characters will be replaced with a double quote (or the character defined by SET NONDISP) and the remainder of the string will be displayed as EBCDIC.

Appendix G. XEDIT Virtual Screens and Windows

General Description

When working on a display terminal, XEDIT always uses a virtual screen and window. The window and virtual screen have the same name. The name depends on whether you specify the WINDOW option on the XEDIT command. If you choose the window option, XEDIT uses the name specified. If no WINDOW option is specified, XEDIT uses "XEDIT" for the virtual screen and window name.

When editing a file, XEDIT checks to see if the specified virtual screen and window exist. If both do not exist, XEDIT attempts to make the window and virtual screen the same size whenever possible. However, windows cannot be larger than the physical screen, and the virtual screen used by XEDIT must have at least 5 lines and 20 columns. XEDIT handles virtual screen and window setup as follows:

1. If neither the window nor the virtual screen exist, XEDIT creates both using the dimensions of the physical screen.
2. If the virtual screen exists and has at least 5 lines and 20 columns, and the window does not exist, XEDIT will define the window to be the same size as the virtual screen. If the virtual screen is smaller than 5 lines by 20 columns, an error message will be given. If the virtual screen is larger than the physical screen, XEDIT adjusts either the lines or columns or both so that the window is not larger than the physical screen (each dimension is adjusted independently).
3. If the window exists but the virtual screen doesn't, the virtual screen will be defined using the window size, if the window is at least 5 lines by 20 columns. If the window is smaller than 5 lines by 20 columns, XEDIT adjusts either the lines or columns or both to match the corresponding dimension(s) of the physical screen (each dimension is adjusted independently).

The following table summarizes these situations:

On input to XEDIT:

Table 3 (Page 1 of 2). How XEDIT Defines Virtual Screens and Windows

Virtual Screen	**Window**	**XEDIT Action**
Does not exist	Does not exist	Virtual screen and window defined with the same dimensions as the physical screen.
Exists	Does not exist	Window defined with the same dimensions of the virtual screen unless the virtual screen is larger than the physical screen. If the virtual screen is larger, then the physical screen size is used to define the window. If the virtual screen is smaller than 5 lines and 20 columns, an error message will be issued.
Does not exist	Exists	Virtual screen defined with the same dimensions of the window unless the window is smaller than 5 lines and 20 columns. If the window is smaller in either dimension or both, XEDIT uses the corresponding dimension(s) of the physical screen to define the virtual screen.

Table 3 (Page 2 of 2). How XEDIT Defines Virtual Screens and Windows

Virtual Screen	Window	XEDIT Action
Exists	Exists	Virtual screen and window will be used as long as the virtual screen has at least 5 lines and 20 columns. If not, error message will be issued.

The following examples illustrate the conditions described in the above table.

- Example 1:

  A virtual screen has 24 lines and 80 columns. A window is not defined. Your terminal has a physical screen with 32 lines and 80 columns. The window will be defined with 24 lines and 80 columns.

- Example 2:

  A virtual screen has 100 lines and 200 columns. A window is not defined. Your terminal has a physical screen with 24 lines and 80 columns. The window will be defined with 24 lines and 80 columns.

- Example 3:

  A window has 20 lines and 60 columns. A virtual screen is not defined. Your terminal has a physical screen with 32 lines and 80 columns. The virtual screen will be defined with 20 lines and 60 columns.

- Example 4:

  A virtual screen has 24 lines and 200 columns. A window is not defined. Your terminal has a physical screen with 32 lines and 80 columns. The window will be defined with 24 lines and 80 columns.

Default Options for the XEDIT Window

In the default situation, that is, when XEDIT defines the window for you, XEDIT uses the following window options:

FIXED
NOBOR
NOPOP
USER
NOTOP

See the *VM/SP CMS User's Guide* for a description of these options.

Full-Screen CMS and the SHOW WINDOW CMSOUT Command

XEDIT will issue the SHOW WINDOW CMSOUT command at various times. If full-screen CMS is ON before XEDIT writes to the screen, SHOW WINDOW CMSOUT will be issued followed by SHOW WINDOW XEDIT or a SHOW for the particular window that has been set up to display the file.

When you return to the XEDIT environment from CMS subset mode, the CMSOUT window will be shown, followed by the window set up to display the file.

If you are already in XEDIT and you set full-screen CMS ON, the CMSOUT window will be shown if the CMSOUT window has never been shown in this XEDIT session and a CMS command is issued, or if an XEDIT message is issued that cannot fit into the XEDIT message area.

Entering XEDIT from Full-Screen CMS

Special Considerations: If you enter XEDIT when full-screen CMS is ON and you then run an application or an exec that issues a prompt and causes a VM READ status, the CMSOUT window will be displayed with a command line. At the command line, type in your response and press the ENTER key. When the exec terminates, the CMSOUT window will be returned to its original state and will be displayed without a command line the next time it is displayed on the screen.

If you enter XEDIT when full-screen CMS is ON and you then run an application or an exec that **does not** issue a prompt but **does** cause a VM READ status, the WM window will be displayed and the following message will be issued:

```
Active window overlaid; enter a windowing command or press a PF key
```

In this situation, exit from the WM window; you will be placed in full-screen CMS and the status line will prompt you:

```
Enter your response in vscreen CMS
```

Enter your response and press the ENTER key. You will again be returned to the WM window and will receive this message:

```
Active window overlaid; enter a windowing command or press a PF key
```

When you exit from the WM window, you will be returned to the XEDIT environment.

Invoking Full-Screen CMS from XEDIT

If you set full-screen CMS ON while you are in the XEDIT environment, the CMS window will be displayed and you will be able to enter commands in the WM window. In this case, the XEDIT window is not visible. To make the XEDIT window visible again, you can use the CMS POP WINDOW command, specifying the window name that XEDIT is using.

Similarly, if you get into a situation where the active window is not large enough for you to enter a command, you can use the PA1 key, which has a default setting of POP WINDOW WM, to get the WM window displayed. You can then enter CMS commands to adjust the window. For example, you can use the CMS MAXIMIZE WINDOW to make the window larger.

Disconnect/Reconnect Considerations

If, while in XEDIT, you disconnect from your terminal and later reconnect to a different type of terminal, the virtual screen and/or window that XEDIT uses may be re-sized to fit the dimensions of the new terminal.

The virtual screen and/or window will change as follows:

- If, before disconnecting, you have **not** defined the virtual screen and window (this is the default situation), then after reconnecting:
  1. Virtual screens are redefined according to the procedure described earlier in this appendix.

2. Windows are resized to fit the new physical screen.

- If you have defined a virtual screen prior to disconnecting, no adjustments will be made to it after reconnecting.
- If you have defined a window prior to disconnecting, and then you reconnect to a **smaller** screen, the window will be adjusted to fit on the new screen. No adjustment is made when reconnecting to an equivalent or larger screen.

Note: If you define the XEDIT virtual screen before disconnecting and then reconnect onto a smaller screen, the XEDIT window may be adjusted such that the resulting window is smaller than the virtual screen. In such a case, the command line may not be visible in the window. Here you can use the PA1 key, which has a default setting of POP WINDOW WM, to get the WM window displayed. You can then use the CMS SCROLL command to position the command line in the window.

Usage Notes

1. On leaving XEDIT, any virtual screens or windows XEDIT defined, including the default XEDIT virtual screen and window, are deleted. Only user-defined virtual screens and windows remain. XEDIT also clears any virtual screens and hides any windows it used but did not create.

   Note that if full-screen CMS is ON and there are multiple XEDIT sessions, the CMSOUT window will be hidden only after leaving the last one.

2. If, during an XEDIT session, you delete the virtual screen that XEDIT is using, XEDIT will redefine the virtual screen according to the guidelines explained in Table 3 on page 441.

3. When a virtual screen is used or has been used by XEDIT, the following CMS commands are invalid for that virtual screen:

   ```
   CLEAR VSCREEN
   CURSOR VSCREEN
   GET VSCREEN
   PUT VSCREEN
   ROUTE
   SET LOGFILE
   SET VSCREEN
   WAITT VSCREEN
   WAITREAD VSCREEN
   WRITE VSCREEN
   ```

   Also, when XEDIT uses a virtual screen, it uses the screen area settings (PRotect/NOProtect, High/NOHigh, color, exthi, psset) defined by XEDIT SET subcommands. These XEDIT-defined settings override any settings that may have been specified on the CMS DEFINE VSCREEN command.

4. If the WINDOW option is specified on the XEDIT command and XEDIT uses an existing virtual screen, then the virtual screen settings will be changed as follows:

   a. The virtual screen TYPE/NOTYPE setting will be set to TYPE.

   b. If logging has been specified for that virtual screen, then it will be set to OFF.

   c. If full-screen CMS is ON, any message classes routed to that virtual screen will be re-routed to the CMS virtual screen.

5. It is possible in certain situations to position your window such that the command line is not visible in the window. To avoid problems, you should establish a method (setting a PF key, for example) to make the command line visible again.

6. If full-screen CMS is ON and abend processing occurs while you are in XEDIT, the CMS window will be displayed with the CMSOUT window popped on top of it. Entering a command will cause the CMSOUT window to be hidden if it is of type SYSTEM or deleted if it is of type USER.

Working in Line-mode

If the XEDIT terminal setting is "TYPEWRITER," then output is written a line at a time to CMS. CMS displays the output based on the terminal type and the full-screen setting.

Under certain circumstances, when you are using the editor in typewriter mode on a display terminal, the CMSOUT window may appear. For example, suppose you are in typewriter mode and you invoke a separate XEDIT session with the CMS XEDIT command, then set full-screen CMS ON during that XEDIT session. Upon exit from full-screen CMS, the CMSOUT window may remain on your screen even though you are now editing in typewriter mode.

Appendix H. A Summary of XEDIT Subcommands and Macros

Subcommand	**Purpose**
Add	Add n line(s) after current line.
ALL	Select a collection of lines for display/editing.
ALter	Change a single character to another (character or hex).
BAckward	Scroll backward n screen displays.
Bottom	Go to last line of file.
CANCEL	Terminate the editing session for all files.
CAppend	Add text to end of current line.
CDelete	Delete characters, starting at column pointer.
CFirst	Move column pointer to beginning of line (zone).
Change	Change one string to another.
CInsert	Insert text starting at the column pointer of the current line.
CLAst	Move the column pointer to the end of the line (zone).
CLocate	Locate a string; move the column pointer and the line pointer.
CMS	Pass a command to CMS, or enter CMS subset mode.
CMSG	Display message in command line of user's screen.
COMMAND	Execute a subcommand without checking for synonym or macro.
COMPress	Prepare line(s) for realignment by replacing blanks with tab characters.
COpy	Copy line(s) at specified location.
COUnt	Display the number of times a string appears.
COVerlay	Replace characters, starting at column pointer.
CP	Pass command to VM/SP control program.
CReplace	Replace characters, starting at the column pointer.
CURsor	Move the cursor to specified position on the screen, and optionally assign a priority for this position.
DELete	Delete line(s) beginning with the current line.
Down	Move line pointer n lines toward end of file (same as NEXT).
DUPlicat	Duplicate line(s).
EMSG	Display a message and sound the alarm.
EXPand	Reposition data according to new tab settings.
EXTract	Return information about internal XEDIT variables and file data.

Subcommand	**Purpose**
FILE	Write file to disk or directory.
Find	Search for line that starts with specified text.
FINDUp	Search for a line that starts with specified text; searches in a backward direction.
FOrward	Scroll forward n screen displays.
GET	Insert lines from another file.
Help	Request online display of XEDIT subcommands and macros; invoke the CMS HELP facility.
HEXType	Display line(s) in hexadecimal and EBCDIC.
Input	Insert a single line, or enter input mode.
Join	Combine two or more lines into one line.
LEft	View data to the left of column one.
LOAD	Read file into storage; use in profile macro only.
Locate	Move line pointer to specified target.
LOWercas	Change uppercase letters to lowercase.
LPrefix	Simulate writing in the prefix area of the current line. Used on typewriter terminals.
MACRO	Execute macro without checking for subcommand or synonym.
MErge	Combine two sets of lines.
MODify	Display a subcommand and its current values in the command line, so it can be overtyped and reentered.
MOve	Move line(s) to another place in the file.
MSG	Display message in message line.
Next	Move line pointer n lines toward end of file (same as DOWN).
NFind	Search forward for first line that does not start with the specified text.
NFINDUp	Search backward for first line that does not start with the specified text.
Overlay	Replace characters in current line.
PARSE	Scan a line of a macro to check the format of its operands.
POWerinp	Enter an input mode for continuous typing.
PREServe	Save settings of XEDIT variables until RESTORE is entered.
PURge	Remove macro from virtual storage.
PUT	Insert lines into another file (new or existing), or into a buffer (to be retrieved by GET from another file).
PUTD	Same as PUT, but delete original lines.

Subcommand	**Purpose**
Query	Display the current value of editing options.
QUIT	End an editing session without saving changes.
READ	Place information from the terminal in the console stack.
RECover	Replace removed lines.
REFRESH	Issued from a macro, it updates the display on the screen.
RENum	Renumber VSBASIC or FREEFORT files.
REPEat	Advance line pointer and reexecute last subcommand.
Replace	Replace current line, or delete current line and enter input mode.
RESet	Remove prefix subcommands or macros when screen is in "pending" status.
RESTore	Restore settings of XEDIT variables to values they had when PRESERVE was issued.
RGTLEFT	Shift display to the right or left; reissue to shift back to original display.
RIght	View data to the right of the last (right-most) column.
SAVE	Write file to disk or directory and remain in edit mode.
SCHANGE	Locate string and make a selective change, using PF keys.
SET ALT	Change the number of alterations that have been made to the file since the last AUTOSAVE and/or since the last SAVE.
SET APL	Inform the editor and CMS if APL keys are used.
SET ARBchar	Define an arbitrary character to be used in a target definition.
SET AUtosave	Automatically issue a SAVE subcommand at specified intervals.
SET BRKkey	Specifies whether or not CP should break in when the "BRKKEY " (defined by CP TERMINAL BRKKEY) is pressed.
SET CASE	Uppercase or lowercase control; specify if case is significant in target searches.
SET CMDline	Move the position of the command line.
SET COLOR	Associate specific colors and attributes with various fields on the XEDIT screen.
SET COLPtr	Specify if column pointer is displayed (typewriter terminals only).
SET CTLchar	Define a control character(s), which associate parts of a reserved line with highlighting, protection, visibility, various colors, extended highlighting, and Programmed Symbol Sets.
SET CURLine	Define the position of the current line on the screen.

Subcommand	Purpose
SET DISPlay	Indicate which selection levels of lines will be displayed on the screen.
SET ENTer	Define a meaning for the ENTER key.
SET ESCape	Define a character that allows you to enter a subcommand while in input mode (typewriter terminals only).
SET ETARBCH	Define an arbitrary character within a file containing Double-Byte Character Set (DBCS) characters to be used in target definitions.
SET ETMODE	Inform the editor that there are Double-Byte Character Set strings in the file.
SET FILler	Define a character that is used when a line is expanded.
SET FMode	Change the file mode of the current file.
SET FName	Change the file name of the current file.
SET FType	Change the file type of the current file.
SET FULLread	Determine whether or not the editor and CMS recognize null characters in the middle of screen lines.
SET HEX	Allows string operands and targets to be specified in hexadecimal.
SET IMage	Control how tabs and backspaces are handled when a line is entered.
SET IMPcmscp	Control whether subcommands not recognized by the editor are transmitted to CMS and CP.
SET LASTLorc	Define the contents of the last locate or change buffer.
SET LINENd	Define a line end character.
SET LRecl	Define a new logical record length.
SET MACRO	Control the order in which the editor searches for subcommands and macros.
SET MASK	Define a new mask, which is the contents of added lines and the input zone.
SET MSGLine	Define position of message line and the number of lines a message may expand to.
SET MSGMode	Control the message display.
SET NONDisp	Define a character to XEDIT and CMS that is used in place of non-displayable characters.
SET NULls	Specify whether trailing blanks are replaced with nulls to allow character insertion.
SET NUMber	Specify whether file line numbers are displayed in the prefix area.
SET PAn	Define a meaning for a PA key.
SET PACK	Specify if the file is to be written in packed format to disk or directory.

Subcommand	Purpose
SET PENDing	Control the execution of a prefix macro and the status of the screen while the macro is being executed.
SET PFn	Define a meaning for a PF key.
SET Point	Define a symbolic name for the current line.
SET PREfix	Control the display of the prefix area or define a synonym for a prefix subcommand.
SET RANge	Define a new "top" and "bottom" for the file.
SET RECFm	Define the record format.
SET REMote	Control the way data transmission is handled in XEDIT and CMS.
SET RESERved	Reserve a line, which cannot be used by the editor.
SET SCALe	Control the display of the scale line.
SET SCOPE	Specify whether the editor operates on the entire file or on only those lines displayed.
SET SCReen	Divide the screen into logical screens, for multiple views of the same or of different files.
SET SELect	Assign a selection level to a line or group of lines in a file.
SET SERial	Control file serialization.
SET SHADow	Specify whether the file is to be displayed with or without shadow lines indicating where lines have been excluded from the display.
SET SIDcode	Specify a character string that is to be inserted into every line of an update file.
SET SPAN	Allows a string target to span a number of lines.
SET SPILL	Control whether or not truncation will occur for certain subcommands.
SET STAY	Specify for certain subcommands whether the line pointer moves when searching for a string.
SET STReam	Specify whether the editor searches only the current line or the whole file for a column-target.
SET SYNonym	Specify whether the editor looks for synonyms; assign a synonym.
SET TABLine	Control the display of the tab line.
SET TABS	Define the logical tab stops.
SET TERMinal	Specify whether a terminal is used in line mode or full-screen mode.
SET TEXT	Inform the editor and CMS if TEXT keys are used.
SET TOFEOF	Control the display of TOF/EOF lines.
SET TRANSLat	Control user-defined uppercase translation.
SET TRunc	Define the truncation column.

Subcommand	Purpose
SET VARblank	Specify whether the number of blanks between two words is significant in a target search.
SET Verify	Control whether lines changed by subcommands are displayed; define the columns displayed and whether displayed in EBCDIC, hexadecimal or both.
SET WRap	Control whether the editor wraps around the file if EOF (or TOF for backwards searches) is reached during a search.
SET Zone	Define new limits within each line for target searches.
SET =	Insert string into the equal buffer.
SHift	Move data right or left (data loss possible).
SI	Continuously add lines and position cursor for indented text.
SORT	Sort all or part of a file, in ascending or descending order.
SOS	Specify functions for screen operation simulation.
SPlit	Split a line into two or more lines.
SPLTJOIN	Split a line or join two lines at the cursor.
STAck	Place line(s) from the file into the console stack.
STATus	Display SET subcommand current settings; create a macro that contains these settings.
TOP	Move line pointer to null TOP OF FILE line.
TRAnsfer	Place editing variable(s) in the console stack, for use by a macro.
Type	Display lines.
Up	Move line pointer n lines toward top of file.
UPPercas	Translate all lowercase characters to uppercase.
Xedit	Edit multiple files.
&	Use before a subcommand to redisplay the command.
=	Reexecute the last subcommand, macro, or CP/CMS command.
?	Display the last subcommand, macro, or CP/CMS command executed.

Prefix	Subcommands
A	Add line(s).
C	Copy line(s).
D	Delete line(s).
E	Extend a line.
F	Move or copy following this line.
I	Insert line(s).

Prefix	**Subcommands**
M	Move line(s).
P	Move or copy preceding this line.
SI	Continuously add lines and position cursor for indented text.
"	Duplicate line(s).
/	Make this line the current line.
SCALE	Display the scale on this line.
TABL	Display the tab line on this line.
.xxxx	Assign symbolic name to this line.
X	Exclude line(s) from display.
S	Show excluded line(s).
<	Shift line(s) to the left.
>	Shift line(s) to the right.

Summary of Changes

Previous editions of this book may be ordered using the pseudo-number found in the *VM/SP Release 6.0 Library Guide, Glossary, and Master Index.*

Summary of Changes
for SC24-5221-05
VM/SP Release 6

The XEDIT enhancements described in this document provide new or improved support in the following areas:

- Shared File System
  - Two new options for the CMS XEDIT command, the XEDIT subcommand, and the LOAD subcommand:

    `LOCK|NOLOCK`
  - A new option for the QUERY subcommand:

    `EDIRNAME`
  - Two new options for the EXTRACT subcommand:

    ```
    LOCK
    EDIRNAME
    ```
  - The format of the EMSG subcommand has been enhanced to accept both three-digit and four-digit message numbers.
- Miscellaneous

  Several minor technical and editorial improvements have been made.

Summary of Changes
for SC24-5221-4
VM/SP Release 5

The XEDIT enhancements described in this document provide new or improved support in the following areas:

- Windowing Support

  A new appendix describing virtual screens and windows.

  A new option for the XEDIT command and the LOAD subcommand:

  `WINDOW`

  Two new options for the EXTRACT subcommand:

  ```
  WINDOW
  NBSCOPE
  ```
- National Language Support (NLS) Enhancements

  NLS Central Message Facility—Most of the XEDIT error messages have been changed in the interest of clarity and usability.

  NLS Parsing Facility—An entry has been added to Appendix A for DLCS (Definition Language for Command Syntax).

- HELP Enhancements

  The user can now use the HELP subcommand to find brief and related information about a command. This is documented under the HELP subcommand.

- Miscellaneous

  Appendix G, "Nondisplayable Character Translation Tables," has been moved to the *VM/SP Terminal Reference.*

Summary of Changes
for SC24-5221-3
VM/SP Release 4

The XEDIT enhancements described in this document provide new or improved support in the following areas:

- Double-Byte Character Set Support (Extended Mode)

  A new appendix describing Double-Byte Character Set Support
  New commands:

  ```
  SET ETARBCH
  SET ETMODE
  ```

  New options for EXTRACT, QUERY, and MODIFY subcommands:

  ```
  ETARBCH
  ETMODE
  ```

- MACLIB support

  New option for XEDIT and LOAD subcommands:

  ```
  MEMBER membername
  ```

  New options for EXTRACT and QUERY subcommands:

  ```
  LIBNAME
  LIBTYPE
  MEMBER
  ```

- Usability Enhancements

  All messages issued by the editor are in mixed case.

- Miscellaneous

  Programmed symbol set support added to:

  ```
  SET COLOR
  SET CTLCHAR
  SET RESERVED
  ```

  EXTRACT CURLINE enhanced
  TASK option added to HELP subcommand
  SI (Structured Input) new subcommand and prefix macro
  SET BRKKEY new subcommand
  File type defaults for BASIC and BASDATA enhanced

Glossary of Terms and Abbreviations

A

abend. (1) Abnormal end of task. (2) Synonym for *abnormal termination.*

abnormal termination. The ending of processing before planned termination. Synonymous with *abend.*

Advanced Program-to-Program Communications/VM (APPC/VM). An API for communicating between two virtual machines that is mappable to the SNA LU 6.2 APPC interface and based on IUCV functions. Along with the TSAF virtual machine, APPC/VM provides this communication within a single system and throughout a collection of systems.

alphameric. Synonym for *alphanumeric.*

alphanumeric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with *alphameric.*

APPC/VM. Advanced Program-to-Program Communications/VM.

assembler language. A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with instruction formats and data formats of the computer.

attention interrupt. An I/O interrupt caused by a terminal user pressing the attention key (or equivalent). See *attention key (ATTN key).*

attention key (ATTN key). A function key on terminals that, when pressed, causes an I/O interruption in the processing unit.

ATTN key. Attention key.

authority. In SFS, the permission to access a file or directory. You can have read authority or write authority (which includes read authority). You can also have file pool administration authority, which is the highest level of authority in a file pool.

AUX file. Auxiliary control file.

B

buffer. An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

C

CMS. Conversational Monitor System.

CMS editor. A CMS facility that lets the user create, change, insert, delete, or rearrange lines of data in a CMS file. See *edit mode* and *input mode.*

CMS EXEC language. A general-purpose, high-level programming language, particularly suitable for EXEC procedures and EDIT macros. The CMS EXEC processor executes procedures and macros (programs) written in this language. Contrast with *EXEC 2 language* and *Restructured Extended Executor (REXX) language.*

CMS files. Refers exclusively to files in the fixed-block format used by CMS file system commands. VSAM and OS data sets and DOS files are not compatible with the CMS file format and cannot be manipulated using CMS file system commands.

CMS file system. A way to create files in the CMS system. CMS files are created by using an identifier consisting of three fields: file name, file type, and file mode. These files are unique to the CMS system and cannot be read or written using other operating systems.

command. A request from a user at a terminal for the execution of a particular CP, CMS, IPCS, GCS, TSAF, or AVS function. A CMS command can also be the name of a CMS file with a file type of EXEC or MODULE. See *subcommand.*

command abbreviation. A short form of the command name, operand, or option that is not a truncation of the word. For example, MSG instead of MESSAGE, RDR instead of READER. Contrast with *truncation.*

command line. The line at the bottom of display panels that lets a user enter commands or panel selections. It is prefixed by an arrow (= = = = >).

component. A collection of objects that together form a separate functional unit. A product may contain many components (for example, VM/SP has components of CP, CMS, GCS, TSAF, IPCS, and AVS). A component can be part of many products. (CP spans both VM/SP and VM/HPO products.)

console stack. Refers collectively to the program stack and the terminal input buffer.

control block. A storage area that a computer program uses to hold control information.

control program. A computer program that schedules and supervises the program execution in a computer system. See *Control Program (CP)*.

Control Program (CP). A component of VM/SP that manages the resources of a single computer so multiple computing systems appear to exist. Each virtual machine is the functional equivalent of an IBM System/370.

Conversational Monitor System (CMS). A virtual machine operating system and component of VM/SP that provides general interactive time sharing, problem solving, program development capabilities, and operates only under the control of the VM Control Program (CP).

CP. Control Program.

CP command. A command available to all VM users. Class G CP commands let the general user reconfigure their virtual machine, control devices attached to their virtual machine, do input and output spooling functions, and simulate many other functions of a real computer console. Other CP commands let system operators, system programmers, system analysts, and service representatives manage the resources of the system.

current line pointer (CLP). A pointer that indicates the line of a CMS file on which the CMS Editor or the System Product Editor (XEDIT) is currently working.

D

DBCS. Double-byte character set.

directory. See *SFS directory*.

disk. A magnetic disk unit in the user's CMS virtual machine configuration. Also called a virtual disk.

disk operating system (DOS). An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data.

display mode. A type of editing at a display terminal in which an entire screen of data is displayed at once and in which the user can access data through commands or by using a cursor. Contrast with *line mode*.

display terminal. A terminal with a component that can display information on a viewing surface such as a CRT or gas panel.

DOS. Disk operating system.

double-byte character set (DBCS). A character set that requires 2 bytes to uniquely define each character. This contrasts with EBCDIC, in which each printed character is represented by 1 byte.

E

edit. A function that makes changes, additions, or deletions to a file on a disk. These changes are interactively made. The edit function also generates information in a file that did not previously exist.

edit mode. The environment in which CMS EDIT subcommands and System Product Editor (XEDIT) subcommands can be entered by the user to insert, change, delete, or rearrange the contents of a CMS file. Contrast with *input mode*.

EOF. End of file.

EXEC 2 language. A general-purpose, high-level programming language, particularly suitable for EXEC procedures and XEDIT macros. The EXEC 2 processor runs procedures and XEDIT macros (programs) written in this language. Contrast with *CMS EXEC language* and *Restructured Extended Executor (REXX) language*.

F

file ID. A CMS file identifier that consists of a file name, file type, and file mode. The file ID is associated with a particular file when the file is created, defined, or renamed under CMS. See *file name*, *file type*, and *file mode*.

file mode. A two-character CMS file identifier field comprised of the file mode letter (A through Z) followed by the file mode number (0 through 6). The file mode letter indicates the minidisk or SFS directory on which the file resides. The file mode number indicates the access mode of the file.

file name. A one-to-eight character alphanumeric field, comprised of A through Z, 0 through 9, and special characters $ # @ + - (hyphen) : (colon) _ (underscore), that is part of the CMS file identifier and serves to identify the file for the user.

file pool. A collection of minidisks managed by SFS. It contains user files and directories and associated control information. Many users' files and directories can be contained in a single file pool.

file space. A user's allocation of space within a file pool.

file type. A one-to-eight character alphanumeric field, comprised of A through Z, 0 through 9, and special characters $ # @ + - (hyphen) : (colon) _ (underscore), that is used as a descriptor or as a qualifier of the file name field in the CMS file identifier. See *reserved file types*.

full-screen CMS. When a user enters the command SET FULLSCREEN ON, CMS is in a window and can take advantage of 3270-type architecture and windowing support, and various classes of output are routed to a set of default windows. Also, users can type commands anywhere on the physical screen and scroll through commands and responses previously displayed. See *windowing*.

I

input line. For typewriter terminals, information keyed in by a user between the time the typing element of the terminal comes to rest following a carriage return until another carriage return is typed. For display terminals, the data keyed into the user input area of the screen. See *user input area*.

input mode. In the CMS Editor or System Product Editor (XEDIT), the environment that lets the user key in new lines of data. Contrast with *edit mode*.

interrupt. A suspension of a process, such as execution of a computer program, caused by an external event and done in such a way that the process can be resumed.

invoke. To start a command, procedure, or program.

L

line end symbol. Synonym for *logical line end symbol*.

line mode. The mode of operation of a display terminal that is equivalent to using a typewriter-like terminal. Contrast with *display mode*.

line number. A number located at either the beginning or the end of a record (line) that can be used during editing to refer to that line.

load. In reference to installation and service, to move files from tape to disk, auxiliary storage to main storage, or minidisks and directories to virtual storage within a virtual machine.

lock. A tool for controlling concurrent usage of SFS objects. Implicit locks are acquired and automatically released when you run CMS commands and program functions in SFS. Explicit locks let you control the type and duration of the lock.

logical line. A command or data line that can be separated from one or more additional command or data lines on the same input line by a logical line end symbol.

logical line end symbol. A special editing symbol, usually the pound (#) sign, that lets the user key in the equivalent of several command or data lines in the same physical line; that is, each logical line except the last line is terminated with the logical line end symbol. Synonymous with *line end symbol*.

M

minidisk. A logical subdivision (or all) of a physical disk pack that has its own virtual device address, consecutive virtual cylinders (starting with virtual cylinder 0), and a VTOC or disk label identifier. Each user virtual disk is preallocated and defined by a VM/SP directory entry as belonging to a user.

module. A unit of a software product that is discretely and separately identifiable with respect to modifying, compiling, and merging with other units, or with respect to loading and execution. For example, the input to, or output from, a compiler, the assembler, the linkage editor, or an exec routine.

N

null line. A logical line with a length of zero that usually signals the CMS Editor to end input mode and enter edit mode. In VM/SP, a null line for typewriter terminals is a terminal input line consisting of a return character as the first and only information, or a logical line end symbol as the last character in the data line. For display devices, a null line is indicated by the cursor positioned at the beginning of the user input area or the data in the user input area ending with a logical line end symbol.

O

operand. Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

P

parameter. A variable that is given a constant value for a specified application and that may denote the application.

password. In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and to gain full or limited access to a system and to the data stored within it.

PF key. Program function key.

physical screen. Synonym for *screen*.

prefix area. The five left-most positions on the System Product Editor's full-screen display, in which prefix subcommands or prefix macros can be entered. See *prefix macros* and *prefix subcommands*.

prefix macros. System Product Editor macros entered in the prefix area of any line on a full-screen display. See *prefix area.*

prefix subcommands. System Product Editor subcommands entered in the prefix area of any line on a full-screen display. See *prefix area.*

program function (PF) key. On a terminal, a key that can do various functions selected by the user or determined by an application program.

program stack. Temporary storage for lines (or files) being exchanged by programs that execute under CMS. See *console stack.*

prompt. A displayed message that describes required input or gives operational information.

prompting. An interactive technique that lets the program guide the user in supplying information to a program. The program types or displays a request, question, message, or number, and the user enters the desired response. The process is repeated until all the necessary information is supplied.

R

read authority. The authority to read the contents of a file without being able to change them. For a directory, read authority lets the user view the names of the objects in the directory.

receive. Bringing into the specified buffer data sent to the user's virtual machine from another virtual machine or from the user's own virtual machine.

reserved file types. File types recognized by the CMS editors (EDIT and XEDIT) as having specific default attributes that include: record size, tab settings, truncation column, and uppercase or lowercase characters associated with that particular file type. The CMS Editor creates a file according to these attributes.

Restructured Extended Executor (REXX) language. A general-purpose programming language, particularly suitable for EXEC procedures, XEDIT macros, or programs for personal computing. Procedures, XEDIT macros, and programs written in this language can be interpreted by the System Product Interpreter. Contrast with *CMS EXEC language* and *EXEC 2 language.*

REXX EXEC. An EXEC procedure or XEDIT macro written in the REXX language and processed by the System Product Interpreter. Synonymous with *REXX program.*

REXX language. Restructured Extended Executor language.

REXX program. Synonym for *REXX EXEC.*

ring of files. The arrangement of files in virtual storage when multiple files are being edited by the System Product Editor.

S

scale. A line on the System Product Editor's (XEDIT) full-screen display, used for column reference.

screen. An illuminated display surface; for example, the display surface of a CRT. Synonymous with *physical screen.*

scrolling. (1) Moving a display image vertically or horizontally in order to view data not otherwise visible within the boundaries of the display screen.
(2) Performing a scroll up, scroll down, scroll right, or scroll left operation.

selective line editing. A feature of XEDIT that allows editing of a specified collection of lines while excluding other lines from the screen.

SFS. Shared file system.

SFS directory. A group of files. SFS directories can be arranged to form a hierarchy in which one directory can contain one or more subdirectories as well as files.

shared file system (SFS). A part of CMS that lets users organize their files into groups known as *directories* and to selectively share those files and directories with other users.

source file. A file that contains source statements for such items as high-level language programs and data description specifications.

source update file. A file containing a single change to a statement in a source file. The file can also include requisite information for applying the change. Synonymous with *update file.*

subcommand. The commands of processors such as EDIT or System Product Editor (XEDIT) that run under CMS.

synonym. In CMS, an alternative command name defined by the user as equivalent to an existing CMS command name. Synonyms are entries in a CMS file with a file type of SYNONYM. Entering the SYNONYM command allows use of those synonyms until that terminal session ends or until the use of synonyms is revoked by entering the SYNONYM command with no operands.

syntax. The rules for the construction of a command or program.

System Product Editor. The CMS facility, comprising the XEDIT command and XEDIT subcommands and macros, that lets a user create, change, and manipulate CMS files.

System Product Interpreter. The language processor of the VM/SP operating system that processes procedures, XEDIT macros, and programs written in the REXX language.

T

target. One of many ways to identify a line to be searched for by the System Product Editor. A target can be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression.

terminal. A device, usually equipped with a keyboard and a display, capable of sending and receiving information.

terminal input buffer. Holds lines entered at the user's terminal until CMS processes them.

truncation. A valid shortened form of CP, CMS, GCS, IPCS, RSCS, TSAF (Query only) command names, operands, and options that can be keyed in. When the shortened form is used, the number of key strokes is reduced. For example, the ACCESS command has a minimum allowable truncation of two, so AC, ACC, ACCE, ACCES, and ACCESS are all recognized by CMS as the ACCESS command.

truncation setting. In the CMS Editor, the value that determines the maximum length of input lines.

typewriter terminal. Printer-keyboard devices that produce hardcopy output only, such as: the IBM 2741 Communication Terminal; the IBM 3215 Console Printer-Keyboard; the IBM 3767 Communication Terminal, Model 1 or 2, operating as a 2741. This term also refers to the IBM 3101 Display Terminal operating as a 2741.

U

update file. Synonym for *source update file*.

user. Anyone who requests the services of a computing system.

user input area. On a display device, the lines of the screen where the user is required to key in command or data lines. See *display mode*, *input line*, and *line mode*.

user-written CMS command. Any CMS file created by a user that has a file type of MODULE or EXEC. Such a file can be executed as if it were a CMS command by issuing its file name, followed by any operands or options expected by the program or EXEC procedure.

V

virtual machine (VM). A functional equivalent of a real machine.

Virtual Machine/System Product (VM/SP). An IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a *real* machine.

virtual screen. A functional simulation of a physical screen. A virtual screen is a *presentation space* where data is maintained. The user can view pieces of the virtual screen through a window on the physical screen.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

VM. Virtual machine.

VM/SP. Virtual Machine/System Product.

vscreen. Virtual screen.

W

window. An area on the physical screen where virtual screen data can be displayed. Windowing lets the user do such functions as defining, positioning, and overlaying windows; scrolling backward and forward through data; and writing data into virtual screens.

windowing. A set of functions that lets the user view and manipulate data in user-defined areas of the physical screen called *windows*. Windowing support lets the user define, position, and overlay windows; scroll backward and forward through data; and write data into virtual screens.

write authority. The authority to read or change the contents of a file or directory. Write authority implies read authority.

X

XEDIT. See *System Product Editor.*

XEDIT macro. (1) A procedure defined by a frequently used command sequence to do a commonly required editing function. A user creates the macro to save repetitious rekeying of the sequence, and invokes the entire procedure by entering a command (that is, the macro file's file name). The procedure can consist of a long sequence of XEDIT commands and subcommands or both, and CMS and CP commands or both, along with REXX or EXEC 2 control statements to control processing within the procedure. (2) A CMS file with a file type of *XEDIT.*

Z

zone-setting. In the CMS editor, a number range that specifies the positions within each data line that can be scanned and edited. In the System Product Editor, the starting position and ending position (columns) of each record within which the editor searches for targets.

Bibliography

Prerequisite Publications:

Virtual Machine/System Product

System Product Editor User's Guide, SC24-5220.

Corequisite Publications:

Virtual Machine/System Product

CMS User's Guide, SC19-6210

System Product Interpreter User's Guide, SC24-5238

System Product Interpreter Reference, SC24-5239

System Product Interpreter Language Reference Summary, SX20-5126.

VM/SP RELEASE 6 LIBRARY

Evaluation

- General Information GC20-1838
- ▨ Introduction GC19-6200
- Introduction to Security SC24-5316

Installation and Service

- ▨ Installation Guide SC24-5237
- ▨ Service Guide SC24-5389

Planning

- ▨ Planning Guide and Reference SC19-6201
- VM Running Guest Operating Systems GC19-6212
- ▨ Release 6 Guide SC24-5368

Operation

- ▨ Operator's Guide SC19-6202

Administration

- ▨ VM System Facilities for Programming SC24-5288
- ▨ Administration SC24-5285
- ▨ Connectivity Planning, Administration, and Operation SC24-5378
- ▨ CMS Shared File System Administration SC24-5367
- ▨ CP System Command Reference SC24-5402

Application Development

- ▨ Application Development Guide for FORTRAN and COBOL SC24-5247
- ▨ Programmer's Guide to the SRPI for VM/SP SC24-5291
- ▨ Application Development Guide for CMS SC24-5286
- ▨ Application Migration Guide for CMS SC24-5366
- ▨ Application Development Reference for CMS SC24-5284
- ▨ Connectivity Programming Guide and Reference SC24-5377
- ▨ GCS Command and Macro Reference SC24-5250
- ▨ **SAA Common Programming Interface Communications Reference** SC26-4399
- ▨ Directory of Programming Interfaces for Customers GC24-5417

Index/Glossary

- ▨ Library Guide and Master Index GC19-6207
- Glossary SC24-5379

End Use

- Quick Reference SX20-4400
- ▨ CMS Primer SC24-5236
- CMS Primer for Line-Oriented Terminals SC24-5242
- ▨ CMS User's Guide SC19-6210
- ▨ CMS Command Reference SC19-6209
- ▨ System Product Editor User's Guide SC24-5220
- ▨ System Product Editor Command and Macro Reference SC24-5221
- ▨ System Product Interpreter User's Guide SC24-5238
- ▨ System Product Interpreter Reference SC24-5239
- ▨ EXEC 2 Reference SC24-5219
- ▨ CP General User Command Reference SC19-6211
- ▨ Terminal Reference GC19-6206

▨ one copy of each shaded manual automatically received with product tape

VM/SP RELEASE 6 LIBRARY

Diagnosis

- System Messages and Codes SC19-6204
- System Messages Cross-Reference SC24-5264
- Service Routines Program Logic LY20-0890
- Interactive Problem Control System Guide and Reference SC24-5260
- Diagnosis Guide LY24-5241
- CP Diagnosis Reference LY20-0892
- CP Data Areas and Control Blocks LY24-5220
- CMS Diagnosis Reference LY20-0893
- CMS Data Areas and Control Blocks LY24-5221
- VM CP Trace Table (Poster) SX24-5225
- Problem Determination Summary SX24-5224

Reference Summaries

- SP Editor Command Language Reference Summary SX24-5122
- EXEC 2 Language Reference Summary SX24-5124
- CMS Primer Summary of Commands SX24-5151
- SP Interpreter Reference Summary SX24-5126
- CMS Primer for Line-Oriented Terminals Summary of Commands SX24-5159
- HELP Facility Introduction SX24-5221
- CP General User Command Reference Summary SX24-5219
- CP System Command Reference Summary SX24-5222
- CMS Command Reference Summary SX24-5220
- VM Summary of End Use Tasks and Commands (Poster) SX24-5173

Auxiliary Communication Support

- VTAM Installation and Resource Definition SC23-0111
- VTAM Customization SC23-0112
- VTAM Operation SC23-0113
- VTAM Messages and Codes SC23-0114
- VTAM Programming SC23-0115
- VTAM Diagnosis Guide LY30-5601
- VTAM Programming for LU 6.2 SC30-3400
- VTAM Data Areas (VM) LY30-5593
- VTAM Reference Summary LY30-5600
- VM/Pass-Through Facility Overview GC24-5373
- VM/Pass-Through Facility: Managing and Using SC24-5374
- RSCS Exit Customization SH24-5197
- RSCS General Information GH24-5055
- RSCS Planning and Installation SH24-5057
- RSCS Messages and Codes SH24-5196
- RSCS Operation and Use SH24-5058
- RSCS Diagnosis Reference LY24-5228
- RSCS Reference Summary SX24-5135

Index

VM/SP
System Product Editor
Command and Macro Reference
Order No. SC24-5221-05

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

______________ ______________ ________ Help Information line ____ of ____

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

____

____

____

____

IBM Branch Office serving you: ____

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

SC24-5221-05

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape Please Do Not Staple Fold and tape

IBM ®

Program Number
5664-167

File Number
S370/4300-39

Printed in U.S.A.

Program Number
5664-167

File Number
S370/4300-39

SC24-5221-05

Printed in U.S.A.