Acknowledgement

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


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

Second Edition (March 1982)

This edition, with Technical Newsletter SN24-5715, applies to release 2 of IBM Virtual Machine/System Product, Program Number 5664-167, and to all subsequent versions and releases until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

Summary of Amendments

For a list of changes, see page iii.

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

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication; if the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Dept. G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1980, 1982
Summary of Amendments
for IBM VM/SP: System Product Editor
Command and Macro Reference

Summary of Amendments
for SC24-5221-1
as updated by SN24-5715
for VM/SP Release 2

New: The SET CTLCHAR subcommand is added, along with
the ability to QUERY/TRANSFER CTLCHAR. The
TAG/NOTAG operand is added to the READ subcommand.
Filetypes NAMES, NETLOG, and NOTEBOOK are assigned
default settings.

Summary of Amendments
for SC24-5221-1
VM/SP Release 1

This edition reflects minor technical changes and editorial
corrections. In addition, the NOCLEAR option has been added
to the XEDIT command.
Publications that support VM/SP as used in conjunction with VM/370 Release 6

Legend:
1. For SNA terminal users, the prerequisite publication is: VM/VCON Installation, Operations, and Terminal Use, Order Number SC27-0602.
2. All users of virtual machine must use the VM/SP System Messages and Codes publication.
3. Contains information on VM/EREP support. EREP Release 3 is recommended for use with VM/SP Release 2.
4. VM/370 Release 8 components. However, the IPCS Extension Program Product (5748-SAI) and the RSCS Networking Program Product (5748-XP1) are recommended for use with VM/SP.
5. If you want all three of the Reference Summary publications, use SBOF 3820 when ordering.

For IBM VM/SP: System Product Editor Command and Macro Reference
Use this publication 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 IBM Virtual Machine/System Product: System Product Editor User's Guide, which also contains information on using the EXEC 2 macro facility 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 sections:

“Section 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 Section 2.

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

“Section 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.

“Section 4: XEDIT Prefix Subcommands” describes the prefix subcommands, which are entered directly in an area called the prefix area of any line in a file.

This publication also has the following appendices:

“Appendix A: Filetype Defaults” lists the special filetypes 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: CMS Editor (EDIT) Compatibility Mode” explains how to edit a file in EDIT compatibility mode.

“Appendix C: Migrating from EDIT to XEDIT” lists the EDIT subcommands and their XEDIT counterparts.

“Appendix D: Display Editing System (EDGAR) Compatibility Mode” explains how to edit a file in EDGAR compatibility mode.

“Appendix E: Migrating from EDGAR to XEDIT” lists the EDGAR subcommands and their XEDIT counterparts.

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

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

Prerequisite Publications:

Corequisite Publications:
IBM Virtual Machine/System Product: EXEC 2 Reference, SC24-5219
Section 1: Rules and Conventions ................................................................. 1-1
The Subcommand Name ............................................................................. 1-1
The Subcommand Operands ...................................................................... 1-1
Character Set Usage .................................................................................. 1-1
Notation Conventions ............................................................................... 1-2

Section 2: The XEDIT Command ................................................................. 2-1

Section 3: XEDIT Subcommands and Macros ............................................. 3-1
ADD ........................................................................................................ 3-2
ALTER (Macro) ...................................................................................... 3-4
BACKWARD ........................................................................................... 3-6
BOTTOM .................................................................................................. 3-7
CANCEL (Macro) .................................................................................... 3-8
CAPPEND (Macro) .................................................................................. 3-9
CDELETE .................................................................................................. 3-10
CFIRST .................................................................................................... 3-12
CHANGE .................................................................................................. 3-13
CINSERT ................................................................................................. 3-16
CLAST ...................................................................................................... 3-17
CLOCATE ............................................................................................... 3-18
CMS ......................................................................................................... 3-21
CMSG ...................................................................................................... 3-23
COMMAND ............................................................................................. 3-24
COMPRESSION ....................................................................................... 3-25
COPY ........................................................................................................ 3-28
COUNT ..................................................................................................... 3-30
COVERLAY ............................................................................................. 3-32
CP ............................................................................................................. 3-33
CREPLACE ............................................................................................... 3-34
CURSOR .................................................................................................. 3-35
DELETE ................................................................................................. 3-37
DOWN .................................................................................................... 3-39
DUPLICAT ............................................................................................. 3-40
EMSG ...................................................................................................... 3-41
EXPAND ................................................................................................. 3-42
FILE ........................................................................................................ 3-43
FIND ......................................................................................................... 3-45
FINDUP ................................................................................................... 3-47
FORWARD ............................................................................................... 3-48
GET ........................................................................................................... 3-49
HELP ....................................................................................................... 3-51
HEXETYPE (Macro) ................................................................................ 3-52
INPUT ...................................................................................................... 3-53
JOIN (Macro) ........................................................................................ 3-55
LEFT ........................................................................................................ 3-57
LOAD ..................................................................................................... 3-59
LOCATE .................................................................................................. 3-62
LOWERCASE ......................................................................................... 3-66
MACRO .................................................................................................... 3-67
MODIFY (Macro) .................................................................................... 3-68
MOVE ..................................................................................................... 3-70
MSG ........................................................................................................ 3-72
NEXT ...................................................................................................... 3-73
NFIND .................................................................................................... 3-75
NFINDUP ............................................................................................... 3-76
OVERLAY ............................................................................................... 3-77
PARSE (Macro) ....................................................................................... 3-78
POWERINP ........................................................................................... 3-80
RESERVE ............................................................................................... 3-82
PRINT ..................................................................................................... 3-84
PUT .......................................................................................................... 3-86
PUTD ....................................................................................................... 3-88
QUERY .................................................................................................... 3-88
QUIT ........................................................................................................ 3-94
READ ...................................................................................................... 3-96
RECOVER ............................................................................................... 3-98
RENUM .................................................................................................. 3-100
REPEAT ................................................................................................. 3-101
REPLACE .............................................................................................. 3-102
RESET ................................................................................................... 3-102
RESTORE ............................................................................................... 3-104
RIGHT .................................................................................................... 3-105
SAVE ...................................................................................................... 3-107
SCHANGE (Macro) .................................................................................. 3-108
SET .......................................................................................................... 3-110
SET APL ................................................................................................. 3-111
SET ARBCHAR ....................................................................................... 3-112
SET Autosave ......................................................................................... 3-114
SET CASE .............................................................................................. 3-116
SET CMDLINE ...................................................................................... 3-117
SET COLPTR .......................................................................................... 3-118
SET CTLCHAR ....................................................................................... 3-118
SET CURLINE ........................................................................................ 3-119
SET ESCAPE ......................................................................................... 3-120
SET FILLER ........................................................................................... 3-121
SET FMODE ........................................................................................... 3-122
SET FNAME .......................................................................................... 3-123
SET FTYPE ............................................................................................ 3-124
SET HEX ................................................................................................. 3-125
SET IMAGE ............................................................................................ 3-126
SET IMPCMSCP ................................................................................... 3-128
SET LINEND .......................................................................................... 3-129
SET LRECL ............................................................................................. 3-130
SET MACRO ........................................................................................... 3-131
SET MASK ............................................................................................. 3-132
SET MSGMODE ....................................................................................... 3-134
SET NONDISP ....................................................................................... 3-135
SET NULLS ............................................................................................ 3-136
SET NUMBER ......................................................................................... 3-137
SET PACK ............................................................................................... 3-138
SET PfN .................................................................................................. 3-139
SET POINT ............................................................................................. 3-141
SET PREFIX ........................................................................................... 3-143
SET RANGE ............................................................................................ 3-144
SET RECFM ........................................................................................... 3-145
SET RESERVED .................................................................................... 3-146
SET SCALE ............................................................................................. 3-148
SET SCREEN .......................................................................................... 3-149
SET SRMIAL ........................................................................................... 3-151
SET SPAN ............................................................................................... 3-153
SET STAY ............................................................................................... 3-155
SET STREAM .......................................................................................... 3-156
SET SYNONYM .................................................................................... 3-157
SET TABLINE ........................................................................................ 3-160
SET TABS ............................................................................................... 3-161
SET TERMINAL ...................................................................................... 3-162
SET TEXT ............................................................................................... 3-163
SET TOFOF ............................................................................................ 3-164
SET TRUNC ............................................................................................ 3-165
SET VARBLANK .................................................................................... 3-166
SET VERIFY ............................................................................................ 3-167
SET WRAP ............................................................................................. 3-169
SET LINE ............................................................................................... 3-170
SET = ...................................................................................................... 3-172
SHIFT ..................................................................................................... 3-173
SORT (Macro) ........................................................................................ 3-174
SOS ......................................................................................................... 3-176
SPLIT (Macro) ........................................................................................ 3-178
STACK ..................................................................................................... 3-180
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status (Macro)</td>
<td>3-181</td>
</tr>
<tr>
<td>Top</td>
<td>3-182</td>
</tr>
<tr>
<td>Transfer</td>
<td>3-183</td>
</tr>
<tr>
<td>Type</td>
<td>3-188</td>
</tr>
<tr>
<td>U / (Ampersand)</td>
<td>3-189</td>
</tr>
<tr>
<td>XEDIT</td>
<td>3-190</td>
</tr>
<tr>
<td>&amp; = (Equal Sign)</td>
<td>3-191</td>
</tr>
<tr>
<td>? (Question Mark)</td>
<td>3-192</td>
</tr>
<tr>
<td>Appendix A: Filetype Defaults</td>
<td>A-1</td>
</tr>
<tr>
<td>Appendix B: CMS Editor (EDIT) Compatibility Mode</td>
<td>B-1</td>
</tr>
<tr>
<td>Appendix C: Migrating from EDIT to XEDIT</td>
<td>C-1</td>
</tr>
<tr>
<td>Appendix D: Display Editing System (EDGAR)</td>
<td>D-1</td>
</tr>
<tr>
<td>Appendix E: Migrating From EDGAR to XEDIT</td>
<td>E-1</td>
</tr>
<tr>
<td>Appendix F: Optimizing Macros</td>
<td>F-1</td>
</tr>
<tr>
<td>Appendix G: A Summary of XEDIT Subcommands and Macros</td>
<td>G-1</td>
</tr>
<tr>
<td>Index</td>
<td>X-1</td>
</tr>
</tbody>
</table>

**Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Character Sets and Their Contents</td>
<td>1-2</td>
</tr>
<tr>
<td>3-1</td>
<td>The ADD Subcommand - Before and After</td>
<td>3-3</td>
</tr>
<tr>
<td>3-2</td>
<td>Realigning a Table</td>
<td>3-26</td>
</tr>
<tr>
<td>3-3</td>
<td>The COPY Subcommand - Before and After</td>
<td>3-29</td>
</tr>
<tr>
<td>3-4</td>
<td>The DELETE Subcommand - Before and After</td>
<td>3-38</td>
</tr>
<tr>
<td>3-5</td>
<td>The LEFT Subcommand - Before and After</td>
<td>3-58</td>
</tr>
<tr>
<td>3-6</td>
<td>The MOVE Subcommand - Before and After</td>
<td>3-71</td>
</tr>
<tr>
<td>3-7</td>
<td>The NEXT Subcommand - Before and After</td>
<td>3-74</td>
</tr>
<tr>
<td>3-8</td>
<td>The RIGHT Subcommand - Before and After</td>
<td>3-106</td>
</tr>
<tr>
<td>3-9</td>
<td>Using the SET MASK Subcommand</td>
<td>3-133</td>
</tr>
<tr>
<td>3-10</td>
<td>The UP Subcommand - Before and After</td>
<td>3-190</td>
</tr>
<tr>
<td>4-1</td>
<td>Prefix Subcommands A and D - Before and After</td>
<td>4-6</td>
</tr>
<tr>
<td>4-2</td>
<td>Prefix Subcommands M and F - Before and After</td>
<td>4-11</td>
</tr>
<tr>
<td>A-1</td>
<td>Default Settings According to Filetype</td>
<td>A-1</td>
</tr>
<tr>
<td>C-1</td>
<td>EDIT Migration Chart</td>
<td>C-1</td>
</tr>
<tr>
<td>E-1</td>
<td>EDGAR Migration Chart</td>
<td>E-1</td>
</tr>
</tbody>
</table>
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 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.

**The Subcommand Operands**

The subcommand operands are keyword and/or positional. 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. The maximum length of a string operand is 160 characters.

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 publication *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 Figure 1-1.
Notation Conventions

The notation used to define the command syntax in this publication 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

  \[ \text{DELe} \text{te} \]

  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 fnf`” indicates that a file identifier such as “MYFILE SCRIPT A1” should be entered.

- **Brackets around a single operand means the operand is optional.**
The "fn" operand is optional.

FILE fn

The "fn" operand must be entered.

• Choices are represented in the format boxes either by stacking the operands or by separating the operands with a vertical bar (|). For example,

A
B
C

or

A | B | C

indicates that you must specify either A, B, or C.

• The use of brackets denotes choices, one of which may be selected. For example,

[A
B
C]

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,

[A
B
C]

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
Section 2: The XEDIT Command

Use the CMS command XEDIT to invoke the editor to create, modify, and manipulate CMS disk files. Once the editor has been invoked, you may execute XEDIT subcommands and use the EXEC 2 macro facility.

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

The format of the XEDIT command is:

```
XEDIT [fn [ft [fm]]] [options...[]]
```

Options:

- [Width nn]
- [NOSCreen]
- [PROFILE macroname]
- [NOPROFil]
- [NOClear]

Options valid only in update mode:

- [Update] [NOUpdate]
- [Seq8] [NOSeq8]
- [Ctl fn1] [NOCtl]
- [Merge]
- [Incr nn]
- [SIDcode string]

where:

- fn ft
  
  are the filename and the filetype 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 filemode of the file to be edited, indicating the disk on which the file resides. The editor determines the filemode of the edited file as follows:
  
  - Editing existing files.
    
    When the filemode is specified, that disk and its extensions are searched. If the filemode is not specified or is specified as an asterisk (*), all accessed disks are searched for the specified file.

  - Creating new files.
    
    If the filemode is not specified, the editor assumes a filemode 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 filetype.

- [NOSCreen]
  
  forces a 3270 display terminal into line (typewriter) mode.
PROFILE macroname

If the specified macro exists on one of the accessed disks, the editor executes it as the first subcommand.

If the specified macro is not found on an accessed CMS disk, 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 filetype 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.

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

Update
The editor searches all accessed CMS disks for a file with a filename of fn and a filetype 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 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 (columns 73-80) contains an eight-digit sequence number in every record of the file to be edited. The SEQ8 option automatically forces the UPDATE option. SEQ8 is the default value.

NOSeq8
specifies that columns 73-75 contain a three-character label field, and that the sequence number is a five-digit number in columns 76-80.

The NOSEQ8 option forces the UPDATE option.

Ctl fn1
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 publication VM/SP: CMS Command and Macro 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.
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 columns 64-71 and pads with blanks, if necessary. (Any data in columns 64-71 is overlaid.) This option forces the UPDATE option.

Usage Notes:
1. For the PROFILE, CTL, SIDCODE, 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".
2. Once the XEDIT command has been executed, the XEDIT subcommand can be used to edit and display multiple files simultaneously (see the XEDIT subcommand).
3. 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.
4. The editor is kept in virtual storage as part of the CMSSEG 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).
5. The following parameters are passed to the PROFILE macro when it is invoked by an XEDIT command:
   • Everything following the command name XEDIT is assigned to the EXEC 2 variable &ARGSTRING.
   • Each parameter following the command name is assigned to an EXEC 2 argument (&1-&n).
   The editor does not examine any parameters that follow a closing right parenthesis on the XEDIT command.
6. When you issue an XEDIT command for a variable-format file, trailing blanks are removed when the file is filed (or saved).
7. 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.

Responses:
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'.
Error Messages:
062E INVALID CHARACTER IN FILEID 'fn ft fm',RC=20
003E INVALID OPTION 'option',RC=24
029E INVALID PARAMETER 'parameter' IN THE OPTION
   'option' FIELD.,RC=24
048E INVALID MODE 'mode',RC=24
054E INCOMPLETE FILEID SPECIFIED.,RC=24
065E 'option' OPTION SPECIFIED TWICE.,RC=24
066E 'option' AND 'option' ARE CONFLICTING OPTIONS.,RC=24
070E INVALID PARAMETER 'parameter',RC=24
002E FILE 'fn ft fm' NOT FOUND.,RC=24
024E FILE 'XEDTEMP CMSUT1 A1' ALREADY EXISTS.,RC=28
069E DISK 'mode' NOT ACCESSSED.,RC=36
229E UNSUPPORTED OS DATA SET.,RC=80,81,82,83
132S FILE 'fn ft fm' TOO LARGE.,RC=88
590E DATA SET TOO LARGE.,RC=88
104S ERROR 'nn' READING FILE 'fn ft fm'
   FROM DISK.,RC=100

Error Messages with UPDATE Options:
007E FILE 'fn ft fm' IS NOT FIXED, 80 CHAR.
   RECORDS.,RC=32
184W './ S' NOT FIRST CARD IN UPDATE FILE -- IGNORED
185W NON NUMERIC CHARACTER IN SEQUENCE
   FIELD '...........',RC=32
186W SEQUENCE NUMBER NOT FOUND.;RC=32
207W INVALID UPDATE FILE CONTROL CARD.,RC=32
174W SEQUENCE ERROR INTRODUCED IN OUTPUT FILE:
   '...........' TO '...........' ,RC=32
179E MISSING OR DUPLICATE 'MACS' CARD IN
   CONTROL FILE 'fn ft fm'
183E INVALID aux/ctl FILE CONTROL CARD.,RC=32
210W INPUT FILE SEQUENCE ERROR '........'
   TO '...........';RC=32
597E UNABLE TO MERGE UPDATES CONTAINING
   './ S' CARDS.,RC=32

Return Codes:
0  Normal
6  Subcommand rejected in the profile due to LOAD error
20  Invalid character in filename or filetype
24  Invalid parameters, or options
28  Source file not found (UPDATE MODE) or specified PROFILE macro does not exist, or file
   XEDTEMP CMSUT1 already exists
32  Error during updating process
36  Corresponding disk not accessed
88  File is too large and does not fit into storage
100 Error reading the file into storage

2 - 4  IBM VM/SP: System Product Editor Command and Macro Reference
Section 3: XEDIT Subcommands and Macros

This section 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 "Section 2: The XEDIT Command."

The editor has two modes of operation: edit mode and input mode. Whenever the XEDIT command is issued, 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 publication 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.
ADD

Use the ADD subcommand to insert blank lines immediately after the current line.
The format of the ADD subcommand is:

```
Add  [n[1]]
```

*where:*

- **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:**
The cursor moves to the first tab column of the first line that was added.
The prefix areas associated with the added lines are highlighted.
The line pointer remains unchanged.

**Error Messages:**
529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE., RC=3
557S NO MORE STORAGE TO INSERT LINES.,RC=4
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,RC=5
588E PREFIX SUBCOMMAND WAITING...,RC=8

**Return Codes:**
- 0 Normal
- 3 Terminal is not a display
- 4 Insufficient storage to add lines
- 5 Invalid operand
- 6 Subcommand rejected in the profile due to LOAD error
- 8 Prefix area contains pending or incomplete subcommand

**Examples:**
Figure 3-1 is a before-and-after example of the ADD subcommand.
ANIMALS FACTS  A1 V 80 TRUNC=60 SIZE=28 LINE=18 COLUMN=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.
== 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.

XEDIT 1 FILE

--- ADD 5

XEDIT 1 FILE

Figure 3-1. The ADD Subcommand - Before and After
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.

The format of the ALTER macro is:

```
ALTER char1 char2 [target [n [p]]]
```

where:

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).

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, one the following message is displayed:

```
517I nn OCCURRENCES CHANGED ON nn LINE(S).
```
Error Messages:
546E TARGET NOT FOUND, RC=2
585E NO LINE(S) CHANGED, RC=4
520E INVALID OPERAND: operand, RC=5
543E INVALID NUMBER: xxx, RC=5
545E MISSING OPERAND(S), RC=5

Return Codes:
0 Normal
1 TOF or EOF reached
2 Target line not found
4 No change occurred
5 Missing or invalid operand
6 Subcommand rejected in the PROFILE due to LOAD error

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.

The format of the BACKWARD subcommand is:

| BACKward | \[n|*|1] |

where:

\( n \) 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.

Error Messages:
529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE,RC=3
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,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
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.

The format of the BOTTOM subcommand is:

```
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).

**Error Message:**

520E INVALID OPERAND : operand,RC=5

**Return Codes:**

- 0 Normal
- 5 Invalid operand
- 6 Subcommand rejected in the profile due to LOAD error
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 issuing a QUIT subcommand for each file.

The format of the CANCEL macro is:

| CANCEL |

Usage Notes:

1. The QUIT that is issued against all the files is either “protected” or “unprotected,” depending on the defined synonyms (see the QUIT subcommand). A protected QUIT causes a warning message to be issued if a file has been modified.

2. 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.

Responses:
(if protected QUIT is defined):

577E FILE HAS BEEN CHANGED. USE QQUIT TO QUIT ANYWAY., RC=12

Error Message:

520E INVALID OPERAND: operand, RC=5

Return Codes:

- 0 Normal
- 5 Invalid operand
- 6 Subcommand rejected in the profile due to LOAD error
- 12 File has been changed (protected QUIT)
CAPPEND (Macro)

Use the CAPPEND macro to append specified text to the end of the current line. The format of the CAPPEND macro is:

```
CAPpend [ text]
```

where:

- `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. Truncation takes place if the appended text goes beyond the truncation column.
2. The text operand starts with the first character following the blank delimiter after the subcommand name.

Responses:

- The column pointer is placed under the first character of the appended text.
- The line pointer remains unchanged.

Error Messages:

- `503E TRUNCATED,RC=3`
- `585E NO LINE(S) CHANGED,RC=4`

Return Codes:

- 0 Normal
- 3 Truncated
- 4 No lines changed
- 6 Subcommand rejected in the profile due to LOAD error

Examples:

Current Line:

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

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....
```

Current Line:

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

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....
```
CDELETE

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

The format of the CDELETE subcommand is:

```
CDelete  [column-target] 1
```  

where:

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. As with all column-targets, the following SET options have an effect on the column-target search:
   
   - SET ARBCCHAR
   - SET CASE
   - SET SPAN
   - SET VARBLANK

2. 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). In this case, several lines may be deleted.

3. Use the CLOCATE subcommand to move the column pointer to the desired location.

Response:

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

```
501I  nn LINES DELETED
```  

Error Messages:

- 546E TARGET NOT FOUND., RC=2
- 585E NO LINE(S) CHANGED., RC=4
- 520E INVALID OPERAND : operand, RC=5
- 588E PREFIX SUBCOMMAND WAITING..., RC=8

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
- 8 Prefix area contains pending or incomplete subcommand
**Examples:**

Current Line:

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

Current Line:

```plaintext
====== A dialect is considered a language if it is used in newspapers.
     |...+...1...+...2...+...3...+...4...+...5...+...6....
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....
```

Current Line:

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

Current Line:

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

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

The format of the CFIRST subcommand is:

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.

Error Message:
520E INVALID OPERAND : operand, RC=5

Return Codes:
0 Normal
5 Invalid operand
6 Subcommand rejected in the profile due to LOAD error

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....

CFIRST

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

The format of the CHANGE subcommand is:

```
Change /string1 [/string2/ [target p q] ]
```

where:

/ (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.

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 publication 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, all characters shifted past the truncation column (see SET TRUNC) will be lost.

   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. 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.

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

4. Using SET STAY and CHANGE:
   If you specify that a change is to occur on multiple lines but string 1 is not found, the current line will be:
   a. unchanged, if SET STAY ON has been issued.
   b. the last line scanned, if SET STAY OFF is in effect (the default).

5. Using SET ZONE and CHANGE:
   The search for string 1 occurs only between the left and right zones. However, characters are shifted or the line is padded with blanks from the left zone up to the truncation column, as explained in usage note 2.

6. The search for string 1 is not affected by the setting of SET SPAN or SET VARBLANK.

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:

```
5171 nn OCCURRENCES CHANGED ON nn LINE(S).
518E nn OCCURRENCES CHANGED ON nn LINE(S);
   nn LINE(S) TRUNCATED.
```

Error Messages:

```
546E TARGET NOT FOUND., RC=2
503E TRUNCATED, RC=3
518E nn OCCURRENCES CHANGED ON nn LINE(S);
   nn LINE(S) TRUNCATED., RC=3
585E NO LINE(S) CHANGED., RC=4
511E STRING2 CONTAINS MORE ARBITRARY CHARACTERS
   THAN STRING1, RC=5
520E INVALID OPERAND : operand, RC=5
543E INVALID NUMBER : xxxxxxxx, RC=5
545E MISSING OPERAND(S), RC=5
```

Return Codes:

```
0 Normal
1 TOP or EOF reached during change
2 Target line not found
3 Truncation occurred during the change
4 No change occurred. (string 1 has not been found)
5 Invalid operand or number
6 Subcommand rejected in the profile due to LOAD error
```
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/  * *

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 immediately before the column pointer. As a result, the data is shifted to the right.

The format of the CINSERT subcommand is:

```
CInsert text
```

where:

*text* is the group of characters to be inserted immediately before 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 issue the CINSERT subcommand without an operand.)

2. Data that is shifted to the right beyond the truncation column will be lost.

3. Use the CLOCATE subcommand to move the column pointer to the desired location.

Responses:

The column pointer and the line pointer remain unchanged.

Error Messages:

- 503E TRUNCATED, RC=3
- 585E NO LINE(S) CHANGED, RC=4
- 545E MISSING OPERAND(S), RC=5

Return Codes:

- 0 Normal
- 3 Truncation occurred
- 4 No line(s) changed
- 5 Missing operand(s)
- 6 Subcommand rejected in the profile due to LOAD error

Examples:

Current Line:

```plaintext
===== 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).

The format of the CLAST subcommand is:

```
CLAST
```

Error Message:

520E INVALID OPERAND : operand, RC=5

Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>5</td>
<td>Invalid operand</td>
</tr>
<tr>
<td>6</td>
<td>Subcommand rejected in the profile due to LOAD error</td>
</tr>
</tbody>
</table>

Examples:

SET ZONE 1 20

Current Line:

```
===== Harvey Kennedy invented the shoelace and made $2.5 million.

|....+....1....+.+.>....+....3....+.+.4....+.+.5....+.+.6....
```

CLAST

```
==== Harvey Kennedy invented the shoelace and made $2.5 million.

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

The format of the CLOCATE subcommand is:

| CLocate | column-target |

where:

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. This 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 this string 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 the string.

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 the left zone (-*) or the right zone (+* or *

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 Right (+) or left (-) search (right is the default)
2 "NOT" symbol (locate something that is not the specified string)
3 character (or hexadecimal) string, enclosed in delimiters
4 "OR" symbol (vertical bar) (locate either one string or another)
5 Up to four strings may be 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 charac-
ter following the column pointer.

CLOCATE /horse/1/buggy/
searches for "horse", then searches for "buggy".

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 subcom-
mand options:

ARBCHAR
CASE
SPAN
VARBLANK

In addition, SET STREAM is meaningful only with column-targets.

STREAM ON
specifies that the search for a string that matches the column-target starts
with the character following (or preceding) the column pointer and continues
to the end (or top) of file.

STREAM OFF
the search is confined to the current line (or zones within the line).

Error Messages:

546E TARGET NOT FOUND.,RC=2
520E INVALID OPERAND: operand,RC=5
545E MISSING OPERAND(S),RC=5
CLOCATE

Return Codes:
0 Normal
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.

Examples:
Current Line:
===== John Keats studied medicine and practiced as an apothecary.
     |...+...1...+...2...+...3...+...4...+...5...+...6...
CLOCATE :6 (absolute column number)
===== John Keats studied medicine and practiced as an apothecary.
     <...|...1...+...2...+...3...+...4...+...5...+...6...
Current Line:
===== James Joyce was a school teacher in Dublin.
     |...+...1...+...2...+...3...+...4...+...5...+...6...
CLOCATE +6 (relative column number)
===== James Joyce was a school teacher in Dublin.
     <...+|...1...+...2...+...3...+...4...+...5...+...6...
Current Line:
===== Herman Melville worked as a customs inspector in N.Y.C.
     |...+...1...+...2...+...3...+...4...+...5...+...6...
CLOCATE /customs/
===== Herman Melville worked as a customs inspector in N.Y.C.
     <...+...1...+...2...+...3...+...4...+...5...+...6...
Current Line:
===== Charles Dickens served as a law clerk and was a reporter.
     |...+...1...+...2...+...3...+...4...+...5...+...6...
CLOCATE /reporter/|/clerk/ (locate "reporter" or "clerk")
===== Charles Dickens served as a law clerk and was a reporter.
     <...+...1...+...2...+...3...+...4...+...5...+...6...
Use the CMS subcommand to force the editor to transmit a command to CMS for execution, or to cause the editor to enter CMS subset mode.

The format of the CMS subcommand is:

```
CMS [ commandline ]
```

where:

`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 with an operand, the editor automatically returns to edit mode after the CMS subset command is executed. As long as the editor is in the shared segment (the default), any CMS command may be executed; otherwise, only the CMS subset commands may be executed.

The following commands may be executed in CMS subset mode:

```
ACCESS  GENMOD  OPTION  SET
ASSGN   GLOBAL  PEEK    START
COMPARE  HELP    PRINT   STATE
CP       IDENTIFY PUNCH   STATEW
DEBUG    INCLUDE  QUERY   SVCTRACE
DISK     LISTFILE RDR     SYNONYM
DLBL     LOAD     RDRLIST TAPE
ERASE    LOADMOD READCARD TELL
FETCH    MODMAP  RECEIVE TYPE
FILEDEF  NAMEFIND RELEASE XEDIT
FILELIST NAMES  RENAME
GENDIRT  NOTE    SENDFILE
```

2. If you enter the CMS subcommand without an operand, you can then use only the CMS subset commands. If you issue any other command you receive the message INVALID SUBSET COMMAND. You must use the CMS subset command RETURN to return to edit mode.

For example:

```
CMS
ERASE MYFILE1 SCRIPT
ERASE MYFILE2 MEMO
RETURN
```

3. 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.

Responses:

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

```
CMS SUBSET
```

This indicates that you are in CMS subset mode. The editor clears the screen before issuing this message; the file display is restored when you enter the RETURN command.

When you issue the CMS subcommand with an operand, and the CMS command does not write on the screen, the file image remains on the screen.
However, if the CMS command displays text, 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.

Error 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
nn The return code of the CMS command after RETURN to XEDIT environment
  0 Normal
  6 Subcommand rejected in the profile due to LOAD error
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.

The format of the CMSG subcommand is:

\[
\text{CMSG} \quad [\text{text}]\]

where:

\text{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 Note:
If it is issued to a typewriter terminal, the CMSG subcommand is a no-op.

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 re-entered.

Responses:
The message is displayed in the command line.

Return Codes:

0 Normal
6 Subcommand rejected in the profile due to LOAD error
COMMAND

Use the COMMAND subcommand to cause the editor to execute a specified XEDIT subcommand without first checking to see if a synonym or macro with the same name exists.

The format of the COMMAND subcommand is:

| COMMAND  | [commandline] |

where:

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.

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

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

Return Codes:

- nn  Return code of the subcommand specified as operand
- 0   Normal
- 6   Subcommand rejected in the profile due to LOAD error
COMPRESS

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

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').

The format of the COMPRESS subcommand is:

| COMPPress         | [target]   |

where:

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 publication 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:
   
   COMPRESS
   SET TABS
   EXPAND

   you can realign an entire table.

2. 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

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

The last line compressed becomes the new current line.

Error Messages:
546E TARGET NOT FOUND.,RC=2
585E NO LINE(S) CHANGED,RC=4
520E INVALID OPERAND : operand,RC=5

Return Codes:
0 Normal
1 TOP 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

Examples:
Figure 3-2 is an example of using the COMPRESS and EXPAND subcommands to realign data in a table.
### Countries with Highest Life Expectancies

<table>
<thead>
<tr>
<th>Country</th>
<th>Men</th>
<th>Women</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sweden</td>
<td>71.8</td>
<td>76.5</td>
</tr>
<tr>
<td>Netherlands</td>
<td>71.0</td>
<td>76.4</td>
</tr>
<tr>
<td>Iceland</td>
<td>70.8</td>
<td>76.2</td>
</tr>
<tr>
<td>Norway</td>
<td>71.0</td>
<td>76.0</td>
</tr>
<tr>
<td>Denmark</td>
<td>70.6</td>
<td>75.4</td>
</tr>
<tr>
<td>Canada</td>
<td>68.7</td>
<td>75.2</td>
</tr>
<tr>
<td>France</td>
<td>68.0</td>
<td>75.5</td>
</tr>
<tr>
<td>Japan</td>
<td>69.0</td>
<td>74.3</td>
</tr>
<tr>
<td>U.K.</td>
<td>68.5</td>
<td>74.7</td>
</tr>
</tbody>
</table>

Figure 3-2. Realigning a Table (Current tab setting is 1 15 20) - COMPRESS (Part 1 of 3)
<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>MEN</th>
<th>WOMEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sweden</td>
<td>71.8</td>
<td>76.5</td>
</tr>
<tr>
<td>Netherlands</td>
<td>71.0</td>
<td>76.4</td>
</tr>
<tr>
<td>Iceland</td>
<td>70.8</td>
<td>76.2</td>
</tr>
<tr>
<td>Norway</td>
<td>71.0</td>
<td>76.0</td>
</tr>
<tr>
<td>Denmark</td>
<td>70.6</td>
<td>75.4</td>
</tr>
<tr>
<td>Canada</td>
<td>68.7</td>
<td>75.2</td>
</tr>
<tr>
<td>France</td>
<td>68.0</td>
<td>75.5</td>
</tr>
<tr>
<td>Japan</td>
<td>69.0</td>
<td>74.3</td>
</tr>
<tr>
<td>U.K.</td>
<td>68.5</td>
<td>74.7</td>
</tr>
</tbody>
</table>

END OF FILE

Figure 3-2. Realigning a Table - Realigned Table (Part 3 of 3)
COPY

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

The format of the COPY subcommand is:

```
COPY target1 target2
```

where:

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 publication 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.
```

Error Messages:

- 505E NOT EXECUTED: THE TARGET LINE (nn) IS WITHIN THE LINES TO COPY.,RC=1
- 546E TARGET NOT FOUND.,RC=2
- 557S NO MORE STORAGE TO INSERT LINES.,RC=4
- 520E INVALID OPERAND: operand,RC=5
- 545E MISSING OPERAND(S),RC=5
- 588E PREFIX SUBCOMMAND WAITING...,RC=8

Return Codes:

- 0 Normal
- 1 Target line within the line to copy
- 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
- 8 Prefix area contains pending or incomplete subcommand

Examples:

Figure 3-3 is a before-and-after example of the COPY subcommand.
00006 SLEEP UNDER WATER.
00007 A LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
00008 ACROSS WATER.
00009 OCTOPUS 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 BOOK LOUSE CAN BE FOUND IN OLD BOOKS. IT FEEDS ON THE GLUE OF THE
1...1...2...3...4...5...6...7...
00016 BINDINGS AND ON HOLD THAT GROW ON OLD PAPER.
00017 ALTHOUGH PORCUPINE FISHES BLOT 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 DRAW 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

Figure 3-3. The COPY Subcommand - Before and After

Section 3:  XEDIT Subcommands and Macros  3 - 29
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.

The format of the COUNT subcommand is:

\[
\text{COUNT} /\text{string}[/\text{target}]\]

where:

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 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 publication 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:
   \[
   \text{SET ARBCHAR ON} \$
   \text{COUNT} /\$(\$)/ *
   \]
   will count all of the expressions enclosed in parentheses.
3. In a macro, you can use TRANSFER LASTMSG to get the number of occurrences.

Responses:
The last line searched becomes the new current line.

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

\[5221 \text{ nn OCCURRENCES}\]

Error Messages:

546E TARGET NOT FOUND, RC=2
520E INVALID OPERAND : operand, RC=5
545E MISSING OPERAND(S), RC=5

Return Codes:

0 Normal
1 TOF or EOF reached during execution
2 Target line not found
5 Missing or invalid operands
6 Subcommand rejected in the profile due to LOAD error
Examples:

Current Line:

AAAA A rose is a rose is a rose.

COUNT/rose/

Response:

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.

The format of the COVERLAY subcommand is:

| COVERlay | text |

where:

- 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.

**Error Messages:**

- 503E TRUNCATED),RC=3
- 585E NO LINE(S) CHANGED,RC=4
- 545E MISSING OPERAND(S),RC=5

**Return Codes:**

- 0 Normal
- 3 Truncation occurred
- 4 No lines changed
- 5 Missing operand(s)
- 6 Subcommand rejected in the profile due to LOAD error

---

**Examples:**

Current Line:

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

CL/summer/  (move the column pointer)

```plaintext
COV winter night?  (blanks are not overlaid)
```

```plaintext
==== Shall I compare thee to a winter's night?  
  <...+....1.....+....2...+....3....+....4....+....5....+....6....
```
Use the CP subcommand to transmit commands to the VM/SP control program environment during an editing session.

The format of the CP subcommand is:

```
CP [ commandline ]
```

where:

- `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.

**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.

However, if the CP command displays text, 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.

**Error Messages:**

513E UNKNOWN CP/CMS COMMAND, RC=-3

**Return Codes:**

-3 Unknown command

nn Return code of the CP command

6 Subcommand rejected in the profile due to LOAD error
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.

The format of theCREPLACE subcommand is:

```
CREPLACE text
```

where:

- **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 the text goes beyond the truncation column, part of the text will be lost, and an error message will be issued.

**Error Messages:**

- 503E TRUNCATED,RC=3
- 585E NO LINE(S) CHANGED,RC=4
- 545E MISSING OPERAND(S),RC=5

**Return Codes:**

- 0 Normal
- 3 Truncation occurred
- 4 No line(s) changed
- 5 Missing operand(s)
- 6 Subcommand rejected in the profile due to LOAD error

---

**Examples:**

**Current Line:**

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

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.....
```
CURSOR

Use the CURSOR subcommand to move the cursor to a specified position.

The format of the CURSOR subcommand is:

```
CURsor    Cmdline   [colno]
          Column
          File     lineno [colno]
          Screen   lineno [colno]
```

where:

- **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.

- **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 screen (under the first character of the prefix area). If not specified, the current column where the cursor is displayed is used.

**Usage Notes:**

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 screen display. For example, in the following subcommand,

   ```
   CURSOR FILE 700
   ```

   file line 700 must appear on the current screen display.

   When you use CURSOR SCREEN lineno, the screen line (lineno) must be within your screen size. Otherwise, the following message is displayed:

   ```
   521E INVALID LINE NUMBER
   ```

2. Remember that the editor uses the first two lines of a screen. Therefore, if you want to move the cursor to the first line of file data, use CURSOR SCREEN 3.

3. The CURSOR COLUMN subcommand is a good candidate for PF key assignment. By default, the editor assigns CURSOR COLUMN to the PF12 key.
4. The CURSOR subcommand with the FILE or SCREEN operands is mainly intended to be issued from XEDIT macros.

5. 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 TRANSFER subcommand).

6. CURSOR COLUMN works only if the current column falls within the first pair of columns defined by the SET VERIFY subcommand.

Error Messages:
521E INVALID LINE NUMBER,RC=1
527E INVALID COLUMN NUMBER,RC=1
529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE,RC=3
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxxx,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 operand
6 Subcommand rejected in the profile due to LOAD error
DELETE

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

The format of the DELETE subcommand is:

```
DELETE [target_1]
```

where:

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 Notes:

You can clear a line by pressing the spacebar once in column one and then pressing the ERASE EOF 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 a line if you press the ERASE EOF key accidentally.

Responses:

If the operand is specified as a string 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 LINES 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.

Error Messages:

546E TARGET NOT FOUND., RC=2
520E INVALID OPERAND : operand, RC=5
588E PREFIX SUBCOMMAND WAITING..., RC=8

Return Codes:

0 Normal
1 Partial delete due to TOP or EOF reached during execution
2 Target line not found
5 Invalid operand
6 Subcommand rejected in the profile due to LOAD error
8 Prefix area contains pending or incomplete subcommand

Examples:

Figure 3-4 is a before-and-after example of the DELETE subcommand.
ANIMALS FACTS AI V 60 TRUNC=80 SIZE=28 LINE=19 COLUMN=1

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, 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.

Figure 3-4. 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.)

The format of the DOWN subcommand is:

| Down | [n] * | 1 |

where:

- \( n \) 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
```

Responses:
The line pointed to becomes the new current line.

Error Messages:

- 520E INVALID OPERAND : operand,RC=5
- 520E INVALID 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

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).

The format of the DUPLICAT subcommand is:

```
DUPLICAT [n [target]]
```

where:

- `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.

Error Messages:
- 546E TARGET NOT FOUND., RC=2
- 557S NO MORE STORAGE TO INSERT LINES., RC=4
- 520E INVALID OPERAND : operand, RC=5
- 543E INVALID NUMBER : xxxxxxxxx, RC=5
- 588E PREFIX SUBCOMMAND WAITING..., RC=8

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
- 8 Prefix area contains pending or incomplete subcommand
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 VM/SP: System Messages). The severity of the message determines whether or not the alarm is sounded.

The format of the EMSG subcommand is:

<table>
<thead>
<tr>
<th>EMSG</th>
<th>[ text]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ mmmnnns text]</td>
</tr>
</tbody>
</table>

where:

text

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

mmmnnnns

is the message identification.

mmm

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

nnn

is the three-digit message number. It may be used to find additional information in the system messages manual (VM/SP: System Messages).

s

indicates the severity and is one of the following:

- R - response
- I - information
- W - warning
- E - error
- S - severe error
- 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. EMSG without operands can 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 VM/SP: CP Command Reference for General Users). In addition, the message is processed according to the SET MSGMODE setting (LONG or SHORT).
4. The message identification (mmmnnnns) 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 screen.

Return Codes:

- 0  Normal
- 6  Subcommand rejected in the profile due to LOAD error
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.

The format of the EXPAND subcommand is:

```
EXPand [target|1]
```

where:

target
defines the number of lines to be expanded, starting with the current line. 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. 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.

Responses:
The last line expanded becomes the new current line.

Error Messages:

- 546E TARGET NOT FOUND., RC=2
- 585E NO LINE(S) CHANGED, RC=4
- 520E INVALID OPERAND : operand, RC=5

Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>TOP or EOF reached during execution</td>
</tr>
<tr>
<td>2</td>
<td>Target line not found</td>
</tr>
<tr>
<td>4</td>
<td>No line(s) changed</td>
</tr>
<tr>
<td>5</td>
<td>Invalid operand</td>
</tr>
<tr>
<td>6</td>
<td>Subcommand rejected in the profile due to LOAD error</td>
</tr>
</tbody>
</table>

Examples:

See the “Examples” section of the COMPRESS subcommand.
FILE

Use the FILE subcommand to write the edited file on disk and, optionally, to override the file identifier originally supplied in the XEDIT command.

The format of the FILE subcommand is:

```
FILE [fn [ft [fm]]]
```

where:

- `fn` indicates the filename for the file. If you do not specify fn, ft and fm cannot be specified, and the existing filename, filetype, and filemode are used.
- `ft` indicates the filetype for the file.
- `fm` indicates the filemode for the file.

Usage Notes:

1. When you specify a file identifier, any existing file that has an identical fileid is replaced. If you change the file identifier and the file being edited had been previously written to disk, that copy of the file is not altered.
2. You can change the filename, filetype, and filemode during an editing session by using the SET FNAME, SET FTYPE, and SET FMODE subcommands.
3. 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.

Notes for Macro Writers:

1. If FILE is issued from a macro, the file is written to disk, 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 and is removed from the set of files being edited.
   - If only one file was being edited, the file is written to disk, that is, a SAVE subcommand is issued for the file; however, you must issue a QUIT when control is returned to the editor.
2. To return control directly to CMS, you can issue the following EXEC 2 statement:
   ```
   &STACK LIFO FILE
   ```
   When the macro completes its execution, control is returned to CMS.

Responses:

If only one file was edited, the CMS ready message indicates that the file has been written to disk, and the user is returned to the CMS environment.

If more than one file was being edited, the current file is written on disk and then is removed from the set of files being edited.

On a typewriter terminal, the following message is displayed:

```
553I EDITING FILE: fn ft fm
```
Error Messages:
520E INVALID OPERAND: operand, RC=5
598S UNABLE TO BUILD UPDATE FILE: INTERNAL LIST DESTROYED, RC=7
599S UNABLE TO BUILD UPDATE FILE: SERIALIZATION DESTROYED, RC=7
037E DISK 'mode' IS READ ONLY, RC=12
531E DISK IS FULL. SET NEW FILEMODE OR CLEAR SOME DISK SPACE, RC=13
062E INVALID CHARACTER IN FILEID 'fn ft fm', RC=20
048E INVALID MODE 'mode', RC=24
069E DISK 'mode' NOT ACCESSED, RC=36

Return Codes:
0 Normal
1 File has been saved and was the only one edited
5 Invalid operand
6 Subcommand rejected in the profile due to LOAD error
7 Error building the update file
12 Disk defined in filemode is read-only
13 Disk is full
20 Invalid character in filename or filetype
24 Invalid filemode
36 Disk not accessed
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.

The format of the FIND subcommand is:

<table>
<thead>
<tr>
<th>Find</th>
<th>text</th>
</tr>
</thead>
</table>

where:

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. Tab characters are converted to blanks (or filler characters) before the search is made if the SET IMAGE ON subcommand has been issued. 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.

Responses:
If text is found, the line containing the specified text becomes the new current line.

Error Messages:
586E NOT FOUND,RC=2
545E MISSING OPERAND(S),RC=5

Return Codes:
0 Normal
2 No line was found
5 Missing operand(s)
6 Subcommand rejected in the profile due to LOAD error
Examples:

Current Line:

====> Dingoes are wild dogs of Australia.

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

====> .

====> 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.

The format of the FINDUP subcommand is:

<table>
<thead>
<tr>
<th>FINDUp</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUp</td>
<td></td>
</tr>
</tbody>
</table>

*where:*

*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. Tab characters are converted to blanks (or filler characters) before the search is made if the *SET IMAGE ON* subcommand has been issued. 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.

*Responses:*

If text is found, the line containing the specified text becomes the new current line.

*Error Messages:*

586E NOT FOUND, RC=2  
545E MISSING OPERAND(S), RC=5

*Return Codes:*

0  Normal  
2  No line was found  
5  Missing operand(s)  
6  Subcommand rejected in the profile due to LOAD error
FORWARD

Use the FORWARD subcommand to scroll toward the end of a file for a specified number of screen displays.

The format of the FORWARD subcommand is:

\[ \text{FORward } [n|*|1] \]

where:

\( n \)

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 Note:
The FORWARD subcommand is assigned by the editor to the PF8 key.

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.

Error Messages:
529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE,RC=3
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,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
6 Subcommand rejected in the profile due to LOAD error
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.

The format of the GET subcommand is:

```
GET [fn [ft [fm] [firstrec [numrec]]]]
```

where:

* fn
  is the filename of the file that contains the data to be inserted into the file you are editing.

* ft
  is the filetype of the file that contains the data to be inserted. If ft is not specified, the filetype of the file you are editing is assumed.

* fm
  is the filemode of the file that contains the data to be inserted. If you do not specify fm, all of your accessed disks 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 filemode, which is optional, you cannot specify any operands that follow. Therefore, if you want to specify firstrec and numrec, you must specify the filename and filetype 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. 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. You should determine how many records were actually copied, and then write the current file on disk.

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

Error Messages:
563W  RECORDS TRUNCATED, RC=3
565W  EOF REACHED; RECORDS TRUNCATED, RC=3
557S  NO MORE STORAGE TO INSERT LINES, RC=4
520E  INVALID OPERAND : operand, RC=5
521E  INVALID LINE NUMBER, RC=5
543E  INVALID NUMBER: xxxxxxxx, RC=5
588E  PREFIX SUBCOMMAND WAITING..., RC=8
062E  INVALID CHARACTER IN FILEID 'fn ft fm', RC=20
048E  INVALID MODE 'mode', RC=24
002E  FILE 'fn ft fm' NOT FOUND., RC=28
562E  NO LINE(S) SAVED BY PUT(D) SUBCOMMAND., RC=28
156E  'RECORD nn' NOT FOUND - FILE 'fn ft fm' HAS ONLY 'nn' RECORDS., RC=32
069E  DISK 'mode' NOT ACCESSED, RC=36
104S  ERROR 'nn' READING FILE 'fn ft fm' FROM DISK, RC=100

Return Codes:
0  Normal
3  Records truncated
4  No more storage available to insert lines
5  Invalid operand or (line) number
6  Subcommand rejected in the profile due to LOAD error
8  Prefix area contains pending or incomplete subcommand
20  Invalid character in filename or filetype
24  Invalid filemode
28  File not found
32  Record “firstrec” is beyond end of file
36  Disk not accessed
100 Error from rdbuf

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, re-appears on the screen.

5. GET (no operands)
   The lines are inserted.
HELP

Use the HELP subcommand: to display a list of all XEDIT subcommands and macros; to display descriptions, formats, and parameters of XEDIT subcommands and macros; or to invoke the CMS HELP command.

The format of the HELP subcommand is:

| Help | [MENU|HELP|name] |

where:

- MENU displays a list of all XEDIT subcommand names and XEDIT macro names.
- HELP displays how to use the XEDIT subcommand HELP.
- name can be any XEDIT subcommand name or XEDIT macro name, and 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, and it must follow the CMS HELP command syntax (refer to the publication VM/SP: CMS Command and Macro Reference).

Usage Notes:
1. XEDIT subcommand or macro abbreviations cannot be used in the name operand. The MENU will provide you with the full name.
2. Use the CMS HELP command format to display information on CMS as well as XEDIT error messages. For example:
   
   HELP DMSnnnt
   
   where nnn is a message number and t is the message type.
3. HELP SET will display a menu for all the SET options.
4. HELP PREFIX will display a menu for all the prefix subcommands.
5. HELP MENU is initially set to the PF1 key.

Error Messages:
From the CMS HELP facility.

Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>6</td>
<td>Subcommand rejected in the profile due to LOAD error</td>
</tr>
<tr>
<td>any number &gt;10</td>
<td>Standard CMS HELP command return codes</td>
</tr>
</tbody>
</table>
HEXTYPE (Macro)

Use the HEXTYPE macro to display a specified number of lines in both hexadecimal and EBCDIC.

The format of the HEXTYPE macro is:

```
HEXTType [ target ]
```

where:

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.
The line pointer moves to the last line typed.

Error Messages:
546E TARGET NOT FOUND., RC=2
520E INVALID OPERAND : operand, RC=5

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

Examples:

Current Line:

```===== To be or not to be -
HEXTYPE  (display the current line in hexadecimal and character)
displays, on a new screen, the following:
E3964082 85409699 409596A3 40A39640 82854060
To be or not to be -
Press the CLEAR key to get the file image back on the screen.```
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.

The format of the INPUT subcommand is:

```
Input [line]
```

where:

- **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 “5731 INPUT MODE:” appears in the message area, and “INPUT-MODE” 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 scale 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.

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.

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, and the screen layout is restored. (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 screen line, you cannot enter data unless you also move the command line (by using the SET CMDLINE subcommand.) You can use SET CMDLINE TOP and SET CMDLINE BOTTOM, but you cannot use SET CMDLINE ON (which assigns the command line to the last two lines on the screen.)
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 keys in input mode, the editor automatically exits from input mode, enters edit mode to execute the subcommand associated with the PF key, and returns to input mode. Therefore, you should carefully select which PF 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 keys are meaningless when used in input mode, for example, a PF key assigned to the ? subcommand.

PF keys that are particularly useful in input mode are those assigned to `TABKEY`, `SPLIT`, and `JOIN`.

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 re-entered automatically. (You cannot use `SET ESCAPE` on a display terminal.)
   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'CIC2C3'`.

**Error Messages:**

503E TRUNCATED, RC=3  
557S NO MORE STORAGE TO INSERT LINES, RC=4  
588E PREFIX SUBCOMMAND WAITING..., RC=8

**Return Codes:**

0 Normal  
3 Truncation occurred  
4 No more space available to add lines  
6 Subcommand rejected in the profile due to LOAD error  
8 Prefix area contains pending or incomplete subcommand
JOIN (Macro)

Use the JOIN macro to merge 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.

The format of the JOIN macro is:

<table>
<thead>
<tr>
<th>Join</th>
<th>Column</th>
<th>CURSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>colno</td>
<td>/string/</td>
</tr>
</tbody>
</table>

where:

no operand joins the current line and the next line. The next line is appended after the first trailing blank in the current line.

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 a good candidate for PF key assignment and is initially assigned by the editor to the PF11 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. In all cases, if the resulting (merged) line is greater than the logical record length, truncation will occur.
2. The original lines that are appended as a result of a JOIN macro are 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.
6. The cursor or the column number or string specified must fall within the current zones.
Error Messages:
585E NO LINE(S) CHANGED,RC=1
561E CURSOR IS NOT ON A VALID DATA FIELD,RC=1
564W EOF REACHED,RC=1
503E TRUNCATED,RC=3
526E OPTION 'CURSOR' VALID IN DISPLAY MODE ONLY,RC=3
520E INVALID OPERAND: operand,RC=5
575E INVALID JOIN COLUMNS DEFINED,RC=5
588E PREFIX SUBCOMMAND WAITING,RC=8

Return Codes:
0 Normal
1 No line(s) changed
3 Truncation occurred
5 Invalid operands
6 Subcommand rejected in the profile due to LOAD error
8 Prefix area contains pending or incomplete subcommands

Examples:
Using JOIN CURSOR Assigned to a PF key:
===== 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 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 column one on the screen. The data starting in column one moves to the right, thus allowing you to see a specified number of positions to its left.

The format of the LEFT subcommand is:

```
LEFT [n|1]
```

where:

- `n` specifies the number of positions to the left of column one 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.
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).
4. The total number of columns shifted cannot exceed the logical record length.

Notes for Macro Writers:
TRANSFER VERSHIFT places the value of `n` (plus or minus) in the console stack so that it can be used by a macro.

Responses:
The data on the screen moves to the right.

Error Messages:
- `520E INVALID OPERAND : operand,RC=5`
- `543E INVALID NUMBER: xxxxxxxx,RC=5`
- `576E 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

Examples:
Figure 3-5 is a before-and-after example of the LEFT subcommand.
Figure 3-5. 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.

The format of the LOAD subcommand is identical to that of the XEDIT command and is as follows:

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

Options:
- [Width nn]
- [NOSCreen]
- [PROFile macroname]
- [NOPROFil]
- [NOCLear]

Options valid only in update mode:
- [Update|NOUpdate]
- [Seg8|NOSeg8]
- [Ctl fn1|NOCtl]
- [Merge]
- [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. The WIDTH option specifies the amount of virtual storage to be used for one file line.
   - If the value of WIDTH specified in the LOAD subcommand is too small to contain a file line, it is overridden by:
     a. the logical record length of the file being read when editing an existing file, or
     b. the WIDTH value (if any) specified in the XEDIT command.
   Therefore, the value of WIDTH specified in the LOAD subcommand cannot cause unwanted truncation.
   WIDTH is the preferred logical record length value for newly-created files.
   - 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 of the file, or
     b. the default logical record length associated with the filetype.
   Note that WIDTH is the only option that has a different meaning in the LOAD subcommand and XEDIT command.

2. 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
LRECL

is the logical record length that is used when the file is written to disk.

TRUNC

is assigned the value of LRECL, except for fixed (F) format files, which have a default value associated with the filetype.

ZONE

is initially set to 1 (zone1) and TRUNC (zone2).

3. 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 and CMS commands can be issued before the LOAD.)

4. 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 filetype. For example:

```plaintext
&IF .&2 NE .&IF &2 NE (&SKIP 3
 &XARGS = &RANGE OF &2 &INDEX
 &STACK LIFO &1 SCRIPT &XARGS
 &READ ARGS
 &LARGS = &RANGE OF &1 &INDEX
 LOAD &LARGS
```

5. 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.

6. 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:
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'.
Error Messages:
508E 'LOAD' MUST BE THE FIRST SUBCOMMAND IN THE PROFILE, RC=3
555E FILE 'fn ft fm' ALREADY IN STORAGE, RC=4
003E INVALID OPTION 'option', RC=24
029E INVALID PARAMETER 'parameter' IN THE OPTION 'option' FIELD, RC=24
048E INVALID MODE 'mode', RC=24
054E INCOMPLETE FILEID SPECIFIED, RC=24
066E 'option' AND 'option' ARE CONFLICTING OPTIONS, RC=24
070E INVALID PARAMETER 'parameter', RC=24
002E FILE 'fn ft fm' NOT FOUND, RC=28
024E FILE 'XEDTEMP CMSUT1 A1' ALREADY EXISTS, RC=28
069E DISK 'mode' NOT ACCESSED, RC=36
229E UNSUPPORTED OS DATA SET, RC=80, 81, 82, 83
132S FILE 'fn ft fm' TOO LARGE, RC=88
590E DATA SET TOO LARGE, RC=88
104S ERROR 'nn' READING FILE 'fn ft fm' FROM DISK, RC=100

Error messages with UPDATE options:
007E FILE 'fn ft fm' IS NOT FIXED, 80 CHAR. RECORDS, RC=32
184W './ S' NOT FIRST CARD IN UPDATE FILE -- IGNORED
185W NON NUMERIC CHARACTER IN SEQUENCE FIELD ' ........', RC=32
186W SEQUENCE NUMBER NOT FOUND, RC=32
207W INVALID UPDATE FILE CONTROL CARD, RC=32
174W SEQUENCE ERROR INTRODUCED IN OUTPUT FILE:
' ........' TO ' ........', RC=32
179E MISSING OR DUPLICATE 'MACS' CARD IN CONTROL FILE 'fn ft fm'.
183E INVALID aux/ctl FILE CONTROL CARD, RC=32
210W INPUT FILE SEQUENCE ERROR ' ........' TO ' ........', RC=32
597E UNABLE TO MERGE UPDATES CONTAINING './ S' CARDS, RC=32

Return Codes:
1 = Normal
3 = LOAD has already been issued
4 = File is already in storage
20 = Invalid character in filename or filetype
24 = Invalid parameters, or options
28 = Source file not found (UPDATE MODE) or the specified profile macro does not exist, or file XEDTEMP CMSUT1 already exists
32 = Error during updating process
36 = Corresponding disk not accessed
88 = File is too large and does not fit into storage
100 = Error reading the file into storage
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.

The format of the LOCATE subcommand is:

```
[Locate] target[ subcommand]
```

where:

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.

4. A string expression defines a group of characters to be located. The charac-
ters 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 exam-
pies, 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/...
```

1 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.
2 “NOT” symbol (Locate something that is not the specified string.)
3 Character (or hexadecimal) string, enclosed in delimiters.
4 “OR” symbol (vertical bar) (Locate one string or another.)
5 Up to four strings may be specified.

For example:

/horse/

searches downward in the file, beginning with 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/

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’.
5. A *complex string expression* has the same format as a simple 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:

**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 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"

**Additional Notes:**

1. **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, the line pointer remains unchanged.

2. **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 top of file).
If SET WRAP ON is in effect, the editor wraps around the file and continues the search for a string expression target up to the line where it started. If a range has been defined, the editor wraps around the bottom of the range and continues at the top of the range.

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.

Notes for Macro Writers
In a macro, an implicit backward locate (for example, -3) is interpreted by EXEC2 as a label. To avoid this problem, use the LOCATE subcommand explicitly (for example, LOCATE -3), or use the COMMAND subcommand (COMMAND -3).

Error Messages:
S46E TARGET NOT FOUND.,RC=2
S20E INVALID OPERAND : operand,RC=5
S45E MISSING OPERAND(S),RC=5

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
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.

The format of the LOWERCAS subcommand is:

```
LOWercas [ target ]
```

where:

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:
When you press the ENTER key, all uppercase letters within the current zones appear in lowercase (if your terminal is equipped with a lowercase feature.)

The last line translated becomes the new current line.

Error Messages:
546E TARGET NOT FOUND.,RC=2
585E NO LINE(S) CHANGED,RC=4
520E INVALID OPERAND : operand,RC=5

Return Codes:
0 Normal
1 TOP 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

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.
MACRO

Use the MACRO subcommand to cause the specified operand to be executed as a macro.

The format of the MACRO subcommand is:

```
MACRO [ macroline ]
```

where:

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 numerals or special characters. It does not follow the usual parsing rule of separating alphabetic characters from immediately following digits or special characters. For example, N2 normally means NEXT 2. MACRO N2 means "execute the macro named N2."

Responses:

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

Error Messages:

542E NO SUCH SUBCOMMAND: 'name'

Return Codes:

nn Return code of the macro specified as operand
    0 Normal
    6 Subcommand rejected in the profile due to LOAD error.

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.
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 re-entered.

The format of the MODIFY macro is:

```
MODIFY keyword
```

where:

`keyword` is one of the following keyword operands valid with the SET, QUERY, or TRANSFER subcommands:

- APL
- FType
- PACK
- SYNonym
- ARBchar
- HEX
- PFn
- TABLine
- AUTosave
- IMage
- Point
- TABS
- CASE
- IMPcmscp
- PREfix
- TERMINal
- CMDline
- LINENd
- RANge
- TEXT
- COLPtr
- LRec1
- RECFm
- TOFOF
- COLumn
- MACRO
- SCALe
- TRunc
- CURLine
- MASK
- SCReen
- VARblank
- ESCape
- MSGMode
- SERial
- Verify
- Filler
- NONDisp
- SPAN
- VERSHift
- FMode
- NULLs
- STAY
- WRap
- FName
- NUMber
- STReam
- Zone

Usage Notes:
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.

Responses:
If the keyword specified is unknown or is an XEDIT variable that cannot be modified, an error message is displayed.

Error Messages:
- 529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE,RC=3
- 545E MISSING OPERAND(S),RC=5
- 520E INVALID OPERAND : operand,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

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.

The format of the MOVE subcommand is:

\[
\text{MOVE target1 target2}
\]

where:

**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 publication VMjSP: 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.

**Error Messages:**
- 505E NOT EXECUTED: THE TARGET LINE (nn) IS WITHIN THE LINES TO MOVE., RC=1
- 546E TARGET NOT FOUND., RC=2
- 509W USE GET AND/OR PUT(D) TO MOVE LINES IN UPDATE MODE., RC=3
- 520E INVALID OPERAND : operand, RC=5
- 545E MISSING OPERAND(S), RC=5
- 588E PREFIX SUBCOMMAND WAITING..., RC=8

**Return Codes:**
- 0 Normal
- 1 Target line within the line to move
- 2 Target line not found
- 3 Move rejected in update-mode
- 5 Invalid or missing operands
- 6 Subcommand rejected in the profile due to LOAD error
- 8 Prefix area contains pending or incomplete subcommand

**Examples:**
Figure 3-6 is a before-and-after example of the MOVE subcommand.
FACTS
V 80 TRUNC=80 SIZE=28 LINE=22 COLUMN=1

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
BINDINGS AND ON MOLDS THAT GROW ON OLD PAPER.
OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISSE LIKE
A SNAKE.
A QUEEN ANT CAN LIVE UP TO 15 YEARS.

---

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.
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.
OSTRICHES CAN KICK LIKE A MULE, IMITATE A LION'S ROAR, AND HISSE LIKE
A SNAKE.

XEDIT 1 FILE

Figure 3-6. The MOVE Subcommand - Before and After
Use the MSG subcommand to display a message in the message area of the screen.

The format of the MSG subcommand is:

\[
\text{MSG} \quad [\text{text}]
\]

where:

\text{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, they are passed to CMS, the screen is cleared, and the messages are displayed. Press the CLEAR key to re-display the file. In this case, the SET CMSTYPE command setting determines whether or not the messages are displayed.

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.
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.)

The format of the NEXT subcommand is:

```
Next [n|*|1]
```

where:

- `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
```

Responses:
The line pointed to becomes the new current line.

Error Messages:
```
520E INVALID OPERAND : operand,RC=5
543E INVALID 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
```

Examples:
Figure 3-7 is a before-and-after example of the NEXT subcommand.
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,
PROFESSOR TWIST COULD NOT BUT SMILE.
"YOU MEAN," HE SAID, "A CROCODILE."

Next 5

Figure 3-7. The NEXT Subcommand - Before and After
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. Only the non-blank characters in the operand are checked against the file. The search starts with the line following the current line.

The format of the NFIND subcommand is:

| NFind | text |

where:

- text is any text, beginning in column one of one or more file lines, that you do not want to find.

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.

Response:
The first line that does not match the operand becomes the new current line.

Error Messages:
- 586E NOT FOUND, RC=2
- 545E MISSING OPERAND(S), RC=5

Return Codes:
- 0 Normal
- 2 No target line was found
- 5 Missing operands
- 6 Subcommand rejected in the profile due to LOAD error
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. Only the non-blank characters in the operand are checked against the file. The search starts with the line preceding the current line.

The format of the NFINDUP subcommand is:

```
NFINDUp  text
NFUp     
```

**where:**
- **text** is any text beginning in column one of one or more file lines, that you do not want to find.

**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.

**Response:**
The first line that does not match the operand becomes the new current line.

**Error Messages:**
- 586E NOT FOUND,RC=2
- 545E MISSING OPERAND(S),RC=5

**Return Codes:**
- 0 Normal
- 2 No target line was found
- 5 Missing operands
- 6 Subcommand rejected in the profile due to LOAD error
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. Replacement starts at the first tab column of the current line.

The format of the OVERLAY subcommand is:

<table>
<thead>
<tr>
<th>Overlay</th>
<th>text</th>
</tr>
</thead>
</table>

where:

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 (filetype ASSEMBLE) whose settings are defined by SET TABS 1 10 16 36 . . . :

```
OVERLAYTTTcomment
```

(where T represents a tab character)

Error Messages:

503E TRUNCATED, RC=3
585E NO LINE(S) CHANGED, RC=4
545E MISSING OPERAND(S), RC=5

Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>3</td>
<td>Truncated</td>
</tr>
<tr>
<td>4</td>
<td>No line(s) changed</td>
</tr>
<tr>
<td>5</td>
<td>Missing operand(s)</td>
</tr>
<tr>
<td>6</td>
<td>Subcommand rejected in the profile due to LOAD error</td>
</tr>
</tbody>
</table>

Section 3: XEDIT Subcommands and Macros 3 - 77
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.

The format of the PARSE macro is:

```
PARSE startcol Alphaword Number String Dblstring Target Word Line
```

where:

- `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.

Usage Notes:

1. Before issuing the PARSE macro, you must place the line to be parsed in the console stack. For example,
   
   ```
   $STACK LIFO $ARGSTRING
   ```

2. As a result of the PARSE macro, several lines are placed 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.
   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:

1. the starting column of the delimited string
2. the length of the delimited string
(3) the starting column of the string itself (without the delimiter)
(4) the length of the string (with delimiters).

A line stacked for a DBLSTRING operand contains:
(1) the starting column of the delimited strings
(2) the length of the delimited strings (including delimiters)
(3) the starting column of the first string itself (without the delimiter)
(4) the length of the first string (without delimiters)
(5) the starting column of the second string (without the delimiter)
(6) 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.

Notes for Macro Writers:
1. The PARSE macro always reads a line from the console stack, if there is one.
2. PARSE always stacks some information upon return, except when operands in the PARSE macro itself are not recognized.
   If the first keyword operand in the PARSE macro does not match the string to be parsed, a single line will be stacked. The line will contain "0", which indicates that no operands were matched and no more lines will be stacked.

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.
6 The subcommand was rejected in the profile macro due to a LOAD error.
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.

The format of the POWERINP macro is:

```
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, restricting them to the proper length by cutting them at word boundaries, if necessary, to make them fit.

4. PF keys cannot be used in power input mode. (Pressing a PF key is equivalent to pressing the ENTER key.)

5. If you want to cause a break in the data that you type in power typing mode, that is, you want data to start on a new line (for example, a new paragraph or SCRIPT/V S 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 typing mode. When characters are inserted, the entire stream of data shifts to the right.

Responses:

In power input mode, the screen changes in the following ways:

- The first line of the screen contains the fileid and the heading,
  
  ```
  * * * POWER_TYPING * * *
  ```

- The second line of the screen contains the current file line in protected format.

- The rest of the screen is blank.
Error Messages:
529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE, RC=3
503E TRUNCATED, RC=4
557E NO MORE STORAGE TO INSERT LINE, RC=4
117S ERROR WRITING TO DISPLAY TERMINAL, RC=8
588E PREFIX SUBCOMMAND WAITING, RC=8

Return Codes:
0 Normal
3 Subcommand valid only for display terminal
4 Truncated or no more storage
6 Subcommand rejected in the profile due to LOAD error
8 I/O error or prefix area contains pending or incomplete subcommand
Use the PRESERVE subcommand to save the settings of various XEDIT variables until a subsequent RESTORE subcommand is issued.

The format of the PRESERVE subcommand is:

```
PREServe
```

**Usage Notes:**

1. The following settings are saved:
   a. Current LEFT or RIGHT
   b. The following SET subcommand options:
      - ARBCHAR
      - AUTOSAVE
      - CASE
      - ESCAPE
      - FILLER
      - FMODE
      - FNAME
      - FTYPE
      - HEX
      - IMAGE
      - LINEND
      - LRECL
      - MACRO
      - MASK
      - MSGMODE
      - NULLS
      - NUMBER
      - PACK
      - PREFIX
      - RECFM
      - SERIAL
      - SPAN
      - STAY
      - STREAM
      - SYNONYM
      - TABS
      - TOFEOF
      - TRUNC
      - VERIFY
      - VARBLANK
      - WRAP
      - ZONE
      - HEX
      - TABS
      - TOFEOF
      - TRUNC
      - VERIFY
      - VARBLANK
      - WRAP
      - ZONE

2. The following values are **not** saved:
   a. Current line pointer
   b. Column pointer
   c. Settings of:
      - APL
      - CMDLINE
      - CURLINE
      - EOF
      - PF string
      - RANGE

**Error Messages:**

- 554E NO STORAGE AVAILABLE, RC=3
- 520E INVALID OPERAND: operand, RC=5

**Return Codes:**

- 0 Normal
- 3 No storage available
- 5 Invalid operands
- 6 Subcommand rejected in the profile due to LOAD error
PURGE

Use the PURGE subcommand to remove the copy of a macro that is in virtual storage.

The format of the PURGE subcommand is:

```
PURge macroname
```

where:

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 reads it into virtual storage and keeps that copy in storage as long as virtual storage is available. (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.

   However, 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. The PURGE subcommand (and the automatic PURGE) cannot be issued from a macro.

Error Messages:

520E INVALID OPERAND : operand, RC=5

Return Codes:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>3</td>
<td>Macro is not currently in storage</td>
</tr>
<tr>
<td>5</td>
<td>Invalid operand</td>
</tr>
<tr>
<td>6</td>
<td>Subcommand rejected in the profile due to LOAD error</td>
</tr>
</tbody>
</table>
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.

The format of the PUT subcommand is:

```
PUT [target] [fn = ft = fm = ]
```

where:

**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.
- 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 filename 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 filetype of the file into which lines are to be inserted. If you omit ft, the editor uses the filetype of the file you are currently editing.

**fm**
- is the filemode of the file into which lines are to be inserted. If you omit fm, the editor uses the filemode 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.

3. If the specified file (fn ft fm) does not already exist, the editor creates it and inserts the lines.

4. If you do not specify a fileid (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 CMS. Thus, issuing a PUT subcommand without specifying a fileid is like storing lines in a temporary holding area. You can retrieve the temporary file without knowing its fileid.

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.

7. The line following the last line PUT (the target line) becomes the new current line.
Response:
If you are creating a file with the PUT subcommand, the following message is displayed.

571I CREATING NEW FILE

For a fixed format file, if a line(s) being PUT is larger than the logical record length of the file in which it is being inserted, the following message is displayed:

579R RECORDS WILL BE TRUNCATED TO nn. CONTINUE (YES/NO)?:

If you enter a NO reply, the following is displayed:

580W PUT NOT EXECUTED

Error Messages:
546E TARGET NOT FOUND.,RC=2
554S NO STORAGE AVAILABLE.,RC=3
520E INVALID OPERAND : operand,RC=5
588E PREFIX SUBCOMMAND WAITING...,RC=8
037E DISK 'mode' IS READ ONLY...,RC=12
062E INVALID CHARACTER IN FILEID 'fn ft fm',RC=20
048E INVALID MODE 'mode',RC=24
069E DISK 'mode' NOT ACCESSED.,RC=36
105S ERROR 'nn' WRITING FILE 'fn ft fm' ON DISK.,RC=100

Return Codes:
0 Normal
1 TOF or EOF reached
2 Target not found; no PUT took place
3 No storage available
5 Invalid operands
6 Subcommand rejected in the profile due to LOAD error
8 Prefix area contains pending or incomplete subcommands
12 Disk is read-only
13 Disk is full
20 Invalid character in filename or filetype
24 Invalid filemode
36 Disk not accessed
100 Error from wrbuf

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.

The format of the **PUTD** subcommand is:

```
```

where:

- **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.
  - 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 filename 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 filetype of the file into which lines are to be inserted. If you omit ft, the editor uses the filetype of the file you are currently editing.

- **fm**
  - is the filemode of the file into which lines are to be inserted. If you omit fm, the editor uses the filemode 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.

For a fixed-format file, if a line(s) being inserted is larger than the logical record length of the file in which it is being inserted, the following message is displayed:

```
579R RECORDS WILL BE TRUNCATED TO nn. CONTINUE (YES/NO)?
```

If you enter a NO reply, the following is displayed:

```
580W PUT NOT EXECUTED
```
Error Messages:
546E TARGET NOT FOUND.,RC=2
554S NO STORAGE AVAILABLE.,RC=3
520E INVALID OPERAND : operand,RC=5
588E PREFIX SUBCOMMAND WAITING...,RC=8
037E DISK 'mode' IS READ ONLY.,RC=12
062E INVALID CHARACTER IN FILEID 'fn ft fm'.,RC=20
048E INVALID MODE 'mode'.,RC=24
069E DISK 'mode' NOT ACCESSED.,RC=36
105S ERROR 'nn' WRITING FILE 'fn ft fm' ON DISK.,RC=100

Return Codes:
0 Normal
1 TOF or EOF reached
2 Target not found; no PUT took place
3 No storage available
5 Invalid operand
6 Subcommand rejected in the profile due to LOAD error
8 Prefix area contains pending or incomplete subcommands
12 Disk is read-only
13 Disk is full
20 Invalid character in filename or filetype
24 Invalid filemode
36 Disk not accessed
100 Error from wrbuf

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.

The format of the QUERY subcommand is:

<table>
<thead>
<tr>
<th>Query</th>
<th>APL</th>
<th>PF[n]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARBchar</td>
<td>POINT [*]</td>
</tr>
<tr>
<td></td>
<td>AUTosave</td>
<td>PRfix</td>
</tr>
<tr>
<td></td>
<td>CASE</td>
<td>RANge</td>
</tr>
<tr>
<td></td>
<td>CMDline</td>
<td>RECFm</td>
</tr>
<tr>
<td></td>
<td>COLPtr</td>
<td>RESERved</td>
</tr>
<tr>
<td></td>
<td>COLumn</td>
<td>RING</td>
</tr>
<tr>
<td></td>
<td>CTLchar [char]</td>
<td>SCALe</td>
</tr>
<tr>
<td></td>
<td>CURLine</td>
<td>SCReen</td>
</tr>
<tr>
<td></td>
<td>CURSor</td>
<td>Seq8</td>
</tr>
<tr>
<td></td>
<td>EOF</td>
<td>SERial</td>
</tr>
<tr>
<td></td>
<td>ESCape</td>
<td>SIDcode</td>
</tr>
<tr>
<td></td>
<td>FILLer</td>
<td>SIZE</td>
</tr>
<tr>
<td></td>
<td>FMode</td>
<td>SPAN</td>
</tr>
<tr>
<td></td>
<td>FName</td>
<td>STAY</td>
</tr>
<tr>
<td></td>
<td>FType</td>
<td>STReam</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>SYNonym [*]</td>
</tr>
<tr>
<td></td>
<td>IMage</td>
<td>TABLine</td>
</tr>
<tr>
<td></td>
<td>IMPcmscp</td>
<td>TABS</td>
</tr>
<tr>
<td></td>
<td>LASTmsg</td>
<td>TARGET</td>
</tr>
<tr>
<td></td>
<td>LENGth</td>
<td>TERMINal</td>
</tr>
<tr>
<td></td>
<td>Line</td>
<td>TEXT</td>
</tr>
<tr>
<td></td>
<td>LINENd</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td>LRecl</td>
<td>TOFEOF</td>
</tr>
<tr>
<td></td>
<td>LScreen</td>
<td>TRunc</td>
</tr>
<tr>
<td></td>
<td>MACRO</td>
<td>UPDATE</td>
</tr>
<tr>
<td></td>
<td>MASK</td>
<td>VARblank</td>
</tr>
<tr>
<td></td>
<td>MSGMode</td>
<td>Verify</td>
</tr>
<tr>
<td></td>
<td>NBFile</td>
<td>VERSShift</td>
</tr>
<tr>
<td></td>
<td>NONDisp</td>
<td>Width</td>
</tr>
<tr>
<td></td>
<td>NULLs</td>
<td>WRap</td>
</tr>
<tr>
<td></td>
<td>NUMBER</td>
<td>Zone</td>
</tr>
<tr>
<td></td>
<td>PACK</td>
<td>=</td>
</tr>
</tbody>
</table>

**where:**

- **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, fileid, and number of alterations.

- **CASE**
  displays the case setting ("U" or "M") and "R" or "I" as defined in the SET CASE subcommand.
CMDline
displays the screen command line number defined by the SET CMDLINE subcommand.

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 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 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 attribute2". If no attributes were defined for the character, displays "CTLCHAR char".

CURLine
displays the line number of the current line relative to the top of the screen, as defined by the SET CURLINE subcommand.

CURSOR
displays the 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 displayed for the position of the cursor in the file.

EOF
displays "ON" or "OFF" as determined by the editor. EOF is "ON" when the line pointer reaches end of file.

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

FILLer
displays the filler character defined by the SET FILLER subcommand.

FMode
displays the two-character filemode.

FName
displays the eight-character filename.

FType
displays the eight-character filetype.

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.

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).
**QUERY**

**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.

**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 physical screen.
  - the number of lines and number of columns of the physical 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.

**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.

**PACK**
- displays “ON” or “OFF” as specified by the SET PACK subcommand.

**PF[n]**
- displays the PF key settings for all the PF keys, or, if n is specified, for a specified PF key number.

**Point [*]**
- QUERY POINT displays the symbolic name associated with the current line, or displays a blank string if no SET POINT has been executed for this line. QUERY POINT * displays all symbolic names that have been defined, starting at the top of the file.

**PREFIX**
- displays “ON” or “OFF” and “RIGHT” or “LEFT” as specified in the SET PREFIX subcommand.

**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.
QUERY

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 area for each file.

The editor displays the following message:
nn FILE(S) IN STORAGE.
The following information is displayed for each file:

```
fn  ft  fm  recfm  Irecl  TRUNC=truneno  SIZE=sizeno
LINE=lineno  COLUMN=colno
```

**where:**

- **fn** is the filename of the file.
- **ft** is the filetype of the file.
- **fm** is the filemode 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.
- **Irecl** is the logical record length of the largest record permitted in the file.
- **truneno** 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.

**SCALE**
- Displays "ON" or "OFF" as specified in the SET SCALE subcommand; if the scale is on the screen, displays the line number of the scale.

**SCREEN**
- Displays the number of lines in each logical screen, in the form “SIZE n1 n2 n3...”, where the first screen uses n1 lines, the second screen uses n2 lines, etc.

**SEQ8**
- Displays "OFF" if the XEDIT command 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.

**SIDCODE**
- Displays the eight-character string specified in the SIDCODE option of the XEDIT command or the LOAD 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.

**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 (that is, 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. (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 n as defined in the SET TABLINE 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: line and column number of the first character in the string; line and column number of the last character in the string.

Displays 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).

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 the top of file.

TOFEOF

displays “ON” or “OFF” as specified in the SET TOFEOF subcommand.

TRunc

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

UPDATE

displays “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

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

3 - 92 IBM VM/SP: System Product Editor Command and Macro Reference
QUERY

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.

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 whatever has been specified in the SET = subcommand.

Notes for Macro Writers:
The QUERY subcommand and the TRANSFER subcommand provide the same list of keyword operands.

Error Messages:
538E NO NAME DEFINED,RC=3
520E INVALID OPERAND : operand,RC=5
525E INVALID PFKEY NUMBER,RC=5
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)
6 Subcommand rejected in the profile due to LOAD error
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. 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.

The format of the QUIT subcommand is:

| QUIT | \[n\] |

where:

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 CMS.

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.

Notes for Macro Writers:

1. QUIT is defined as a synonym to PQUIT. The PQUIT (protected quit) subcommand causes the warning message to be displayed. 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 only one file was being edited, a QUIT issued from a macro has no effect. When control is returned to the editor, you must issue a QUIT.

3. To return control directly to CMS, you can issue the following EXEC 2 statement:

   &STACK LIFO QUIT

   When the macro completes its execution, control is returned to CMS.

   If a return code is specified, for example, &STACK LIFO QUIT n, the return code becomes the XEDIT command return code.

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

Error Messages:
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,RC=5

Return Codes:
0 Normal
1 Only one file was edited
5 Invalid operand or number
6 Subcommand rejected in the profile due to LOAD error
12 File has been changed. Use QUIT to QUIT anyway.
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, the macro has access to the stacked information via the EXEC 2 &READ control statement, for example.

The format of the READ subcommand is:

| READ | Cmdline | All [NUMBER] | TAG | Nochange [NUMBER] | NOTAG |

where:

Cmdline
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.

NOTAG
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.

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 a PF key or the ENTER key. Pressing the CLEAR, PA1, or PA2 key does not terminate a READ. (Pressing the CLEAR key clears any data that was typed since the last time the ENTER key or a PF key was pressed; the PA1 key forces CP mode; and the PA2 key is ignored.)

4. Which lines are placed in the console stack depends on:
   • which operand is specified (Cmdline, All, or Nochange)
   • whether the READ was terminated by pressing a PF key or the ENTER key.

Pressing a PF key causes the current value of that PF key to be stacked. (However, if a PF key is set to COPYKEY or TABKEY, the PF key value is not stacked.) The command line is always stacked as the last line, unless the PF key is set to the ? subcommand, in which case the command line is not stacked. (The same 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 5, below.)
Using READ CMDLINE

If the command line was changed, the console stack may contain two lines:

a. PF key value (if a PF key was pressed)
b. command line

(If the command line was changed and the ENTER key was pressed, only the command line is stacked.)

If nothing was entered on the command line and the ENTER key is pressed, one of the following is stacked:

- a null line (if NOTAG was specified)
- the tag “CMD” followed by a null string (if TAG was specified)

Any fields defined within reserved lines (lines reserved by SET RESERVED) are ignored. Any prefix subcommands are executed.

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. PF key value (if a PF key was pressed)
b. lines and prefix areas changed on the screen (if any)
c. command line

If nothing was changed on the screen and the ENTER key is pressed, one of the following is stacked:

- a null line (if NOTAG was specified)
- the tag “CMD” followed by a null string (if TAG was specified)

5. 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
- FIL - file line
- 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 the ENTER key is pressed without entering anything in the command line, or if a command is suppressed by using the ERASE EOF key. In this case, only the tag (CMD) is stacked.
- 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. n3 is returned only if the READ subcommand was issued with the NUMBER option.
READ

- string
  is the changed file line.  (String can be empty if the ERASE EOF key was pressed.)

- 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.  n3
  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.)

Responses:
The message, "MACRO-READ", is displayed in the status area.

Error Messages:
520E INVALID OPERAND : operand,RC=5

Return Codes:
0 Normal
5 Invalid operand
6 Subcommand rejected in the profile due to LOAD error
RECOVER

Use the RECOVER subcommand to replace a specified number of lines that were removed by a DELETE or a PUTD subcommand, or a D (delete) prefix subcommand.

The format of the RECOVER subcommand is:

```
RECover [n] *[1]
```

where:

- `n` is the number of lines removed by a DELETE or 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. 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.
2. 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.
3. RECOVER subcommands do not have a one-to-one correspondence with DELETE subcommands. The following is a valid sequence:

```
DELETE 2
.
.
DELETE 3
.
.
RECOVER 1
.
.
RECOVER 4
```

4. The size of the recoverable lines buffer depends on the amount of virtual storage.

5. If you are editing a file in update mode (where the status area indicates UPDATE-MODE), no lines can be recovered. Update mode is in effect when one of the following options is specified with the XEDIT command: UPDATE, SEQ8, NOSEQ8, CTL, MERGE, INCR.

Response:

The line(s) that is recovered is inserted just before the current line. The following message is displayed:

```
5021 nn LINE(S) RECOVERED.
```

Error Messages:

```
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,RC=5
588E PREFIX SUBCOMMAND WAITING...,RC=8
```
Return Codes:
0  'n' lines have been inserted
3  Pool of deleted line is empty
5  Invalid operand or number
6  Subcommand rejected in the profile due to LOAD error
8  Prefix area contains pending or incomplete subcommand
RENUM

Use the RENUM subcommand to renumber the line numbers of VSBASIC or FREEFORT files.

The format of the RENUM subcommand is:

```
RENum [startno 10 [incr]]
```

where:

- `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 compatibility mode; XEDIT does not perform line number editing.

Error Messages:
- 520E INVALID OPERAND : operand,RC=5
- 535E INVALID PARMS FOR RENUM.,RC=5
- 543E INVALID NUMBER : xxxxxxxx,RC=5
(See also EDIT messages.)

Return Codes:
- 0 Normal
- 5 Invalid operands
- 6 Subcommand rejected in the profile due to LOAD error
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.

The format of the REPEAT subcommand is:

```
REPEat [target|1]
```

where:

target

specifies the number of lines that the REPEAT subcommand executes upon, starting on the line following the current line, up to but not including the target line. 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
   =

Response:

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

Error Messages:

546E TARGET NOT FOUND., RC=2
520E INVALID OPERAND: operand, RC=5

Return Codes:

nn Same as the repeated subcommand’s return codes
1 TOF or EOF reached
2 Target not found
5 Invalid operand
6 Subcommand rejected in the profile due to LOAD error
REPLACE

Use the REPLACE subcommand to replace the current line with a specified line or to delete the current line and enter input mode.

The format of the REPLACE subcommand is:

| Replace | [text] |

where:

- 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 Note:

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.

Responses:

When you issue a REPLACE subcommand with no operand, the following message is displayed:

573I INPUT MODE:

Error Messages:

- 503E TRUNCATED,RC=3
- 557S NO MORE STORAGE TO INSERT LINES.,RC=4
- 588E PREFIX SUBCOMMAND WAITING...,RC=8

Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>3</td>
<td>Line was truncated</td>
</tr>
<tr>
<td>4</td>
<td>Not enough storage to accomodate added lines</td>
</tr>
<tr>
<td>6</td>
<td>Subcommand rejected in the profile due to LOAD error</td>
</tr>
<tr>
<td>8</td>
<td>Prefix area contains pending or incomplete subcommand</td>
</tr>
</tbody>
</table>

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 when the screen is in a “pending” or “incomplete” status.

The format of the RESET subcommand is:

```
RESET
```

Usage Notes:

1. The RESET subcommand removes all prefix subcommands when the screen displays a “COPY/MOVE PENDING” or “BLOCK INCOMPLETE” status.

2. To remove prefix subcommands when the screen is not in a pending or incomplete status, press the CLEAR key.

Response:

The original prefix area (equal signs or line numbers) replaces any prefix subcommands that were typed in.

Error Messages:

520E INVALID OPERAND : operand,RC=5

Return Codes:

- 0 Normal
- 5 Invalid operands
- 6 Subcommand rejected in the profile due to LOAD error

Examples:

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.
Use the RESTORE subcommand to restore the settings of the XEDIT variables to the values they had when the PRESERVE subcommand was last issued.

The format of the RESTORE subcommand is:

```
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.

**Error 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
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.

The format of the RIGHT subcommand is:

```
RIGHT [n|1]
```

where:

- **n** specifies the number of positions to the right of the last column 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.
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 shifted cannot exceed the logical record length.

Notes for Macro Writers:

Use TRANSFER VERSHIFT to place the value of n (plus or minus) in the console stack so that it can be used by a macro.

Response:
The data on the screen moves to the left.

Error Messages:

- 520E INVALID OPERAND : operand,RC=5
- 543E INVALID NUMBER: xxxxxxxx,RC=5
- 576E 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

Examples:

Figure 3-8 is a before-and-after example of the RIGHT subcommand.
THE SONG OF CANARIES
NEVER VARIES.
AND WHEN THEY'RE MOURNING
THEY'RE PRETTY REVOLTING.

THE GIRAFFE

I BEG YOU, CHILDREN, DO NOT LAUGH
WHEN YOU SURVEY THE TALL GIRAFFE.

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 GIRAFFE LIFE.
** END OF FILE **

Figure 3-8. The RIGHT Subcommand - Before and After
SAVE

Use the SAVE subcommand to write the file that is currently being edited on disk without returning control to CMS, and, optionally, to change the file identifier.

The format of the SAVE subcommand is:

```
SAVE  [ fn  [ ft  [ fm ] ] ]
```

where:

- \( fn \)
  indicates the filename of the file to be saved. If you specify only \( fn \), the filetype and filemode remain the same.

- \( ft \)
  indicates the filetype of the file to be saved.

- \( fm \)
  indicates the filemode of the file to be saved.

**Usage Notes:**

1. If you specify a new file identifier, the original copy of the file on disk is not changed. Changing the file identifier simply creates another copy of the file with a new file identifier.

2. If you specify a new file identifier, any existing file with the same file identifier is replaced; no message is issued.

3. To write a file on disk and end the editing session, use the FILE subcommand instead of the SAVE subcommand.

4. If you want a file to be saved automatically at regular intervals, use the SET AUTOSAVE subcommand.

5. 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.

6. If the automatic save function is in effect (SET AUTOSAVE subcommand), the corresponding AUTOSAVE file is erased when a SAVE is executed.

**Error Messages:**

- 520E INVALID OPERAND : operand,RC=5
- 598S UNABLE TO BUILD UPDATE FILE : INTERNAL LIST DESTROYED.,RC=7
- 599S UNABLE TO BUILD UPDATE FILE : SERIALIZATION DESTROYED.,RC=7
- 037E DISK 'mode' IS READ ONLY.,RC=12
- 531E DISK IS FULL. SET NEW FILEMODE OR CLEAR SOME DISK SPACE.,RC=13
- 062E INVALID CHARACTER IN FILEID 'fn ft fm',RC=20
- 048E INVALID MODE 'mode',RC=24
- 069E DISK 'mode' NOT ACCESSED.,RC=36

**Return Codes:**

- 0 Normal
- 5 Invalid operand
- 6 Subcommand rejected in the profile due to LOAD error
- 7 Error building the update file
- 12 Disk defined in filemode is read-only
- 13 Disk is full
- 20 Invalid character in filename or filetype
- 24 Invalid filemode
- 36 Disk not accessed
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 only when it has been assigned to a PF key and only when a CHANGE or CLOCATE subcommand (in the format CLOCATE/string/) has been typed in the command line.

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.)

The format of the SCHANGE macro is:

| SCHANGE | [keynumber] |

where:

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.

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.

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 and the first PF key is pressed, the following message is displayed when a string is located:

551I STRING string1 FOUND.

If a CHANGE subcommand is typed in the command line and the first PF key is pressed, the following message is displayed when a string is located:

551I STRING string1 FOUND. --- PFxx 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, the following message is displayed:

```
569E NO "CHANGE" OR "CLOCATE"
SUBCOMMAND SPECIFIED,RC=5
```

**Error Messages:**

- 561E CURSOR IS NOT ON A VALID DATA FIELD
- 529E SUBCOMMAND IS ONLY VALID IN DISPLAY MODE,RC=1
- 586E NOT FOUND ON SCREEN,RC=2
- 520E INVALID OPERAND: operand,RC=5
- 525E INVALID PFKEY NUMBER,RC=5
- 545E MISSING OPERAND(S),RC=5
- 574E CHANGE NOT VALID {WITH CLOCATE|AFTER CURSOR MOVEMENT}

**Return Codes:**

- 0 Normal
- 1 Subcommand valid only in display mode
- 2 Not found
- 5 Invalid or missing operands
- 6 Subcommand rejected in the profile due to LOAD error
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 below. A complete description of each option follows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APL</td>
<td>LRecl</td>
<td>SPAN</td>
</tr>
<tr>
<td>ARBchar</td>
<td>MACro</td>
<td>STAY</td>
</tr>
<tr>
<td>Autosave</td>
<td>MASK</td>
<td>STream</td>
</tr>
<tr>
<td>CASE</td>
<td>MSGMode</td>
<td>SYNonym</td>
</tr>
<tr>
<td>CMDline</td>
<td>NONDisp</td>
<td>TABLine</td>
</tr>
<tr>
<td>COLPtr</td>
<td>NULLs</td>
<td>TABS</td>
</tr>
<tr>
<td>CTLchar</td>
<td>NUMBER</td>
<td>TERMINal</td>
</tr>
<tr>
<td>CURLine</td>
<td>PACK</td>
<td>TEXT</td>
</tr>
<tr>
<td>ESCape</td>
<td>PFn</td>
<td>TOFEOF</td>
</tr>
<tr>
<td>Filler</td>
<td>Point</td>
<td>TRunc</td>
</tr>
<tr>
<td>FMode</td>
<td>PREFix</td>
<td>VARblank</td>
</tr>
<tr>
<td>FName</td>
<td>RANGE</td>
<td>Verify</td>
</tr>
<tr>
<td>FType</td>
<td>RECFm</td>
<td>WRap</td>
</tr>
<tr>
<td>HEX</td>
<td>RESERved</td>
<td>Zone</td>
</tr>
<tr>
<td>IMAGE</td>
<td>SCALe</td>
<td></td>
</tr>
<tr>
<td>IMPcmscp</td>
<td>SCReen</td>
<td></td>
</tr>
<tr>
<td>LINENd</td>
<td>SERial</td>
<td></td>
</tr>
</tbody>
</table>

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 APL

Use the APL option to inform the editor if APL keys are available on the terminal. The format of the SET APL subcommand is:

| [SET] | APL ON | OFF |

where:

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:
according to CP TERMINAL APL setting when the editor is invoked

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. In CP mode, you can issue a TERMINAL APL ON command, which will be recognized when the editor is invoked.
   b. If the CP TERMINAL command has not been issued in CP mode, you must issue the editor SET APL subcommand.

If instead, you issue a TERMINAL command directly to CP while in edit mode (via the CP or CMS subcommand or in subset mode), the editor will not recognize the command and will not perform the necessary conversions.

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.

Error Messages:
524W NONDISP CHARACTER RESET TO BLANK.
526E OPTION 'APL' VALID IN DISPLAY MODE ONLY.,RC=3
520E INVALID OPERAND : operand,RC=5
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
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."

The format of the SET ARBCHAR subcommand is:

| [SET] | ARBchar ON [char] | OFF |

where:

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 $

Usage Notes:
1. Arbitrary characters can be used in targets and in the CHANGE subcommand.
2. The arbitrary character may be specified in hexadecimal if SET HEX ON is in effect.

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. LOCATE /($)/
   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 ARBCCHAR:

String2 must not contain more arbitrary characters than string1. If it does, the following error message is displayed:

511E STRING2 CONTAINS MORE ARBITRARY CHARACTERS THAN STRING1, RC=5

For example:

<table>
<thead>
<tr>
<th>Subcommand:</th>
<th>SET ARBCCHAR ON $</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Line:</td>
<td>the white house with several large windows</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subcommand:</th>
<th>C /the$house$windows/a$farmhouse$two$shutters/</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>error message</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subcommand:</th>
<th>C /the$house$windows/a$farmhouse$shutters/</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>a white farmhouse with several large shutters</td>
</tr>
</tbody>
</table>

String2 can have fewer arbitrary characters than string1. For example:

<table>
<thead>
<tr>
<th>File Line:</th>
<th>the white house with several large windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcommand:</td>
<td>C/the$house$windows/a$large farmhouse/</td>
</tr>
<tr>
<td>Result:</td>
<td>a white large farmhouse</td>
</tr>
</tbody>
</table>

Special use of $ as 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 column1.)
Example:
CHANGE /$A/A/  
deletes all characters that precede A.
```

```
string1: /B$/  
The $ means, "the string starting with the character following B and ending at the truncation column."
Example:
CHANGE /B$/B/  
deletes the characters that follow B.
```

Error 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
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.

The format of the SET AUTOSAVE subcommand is:

```
[SET] AUTosave n [mode] OFF
```

where:

- **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 mini-disk on which the file is written. The default is the A-disk.

Initial Setting:

AUTOSAVE OFF

Usage Notes:

1. You can use the QUERY AUTOSAVE subcommand to display the current AUTOSAVE setting.

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 rename the AUTOSAVE file and resume editing; the file will reflect all changes made up to the time of the last AUTOSAVE.

4. To recover from an erroneous change to the file, for example, a bad global change, you should immediately issue a QUIT subcommand, erase the file, rename the AUTOSAVE file, and continue.

Notes for Macro Writers:

You can use the TRANSFER AUTOSAVE subcommand to stack the autosave information for further processing by a macro.

Responses:

Each time an automatic save occurs, the editor writes the file on your A-disk (or on the disk specified in the SET AUTOSAVE subcommand). It assigns the file a unique eight-digit filename and a filetype of AUTOSAVE.

If no other AUTOSAVE file exists, the filename and filetype are:

1 AUTOSAVE

If an AUTOSAVE file exists, the filename is the next available number.

Error Messages:

520E INVALID OPERAND: operand,RC=5
545E MISSING OPERAND(S),RC=5
037E DISK 'mode' IS READ ONLY,RC=12
048E INVALID MODE 'mode',RC=24
069E DISK 'mode' NOT ACCESSED.,RC=36
Return Codes:

0 Normal
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error.
12 Disk is read only
24 Invalid filemode
36 Corresponding disk not accessed
SET CASE

Use the CASE option to control how letters are entered into the file and to specify if uppercase and lowercase letters are to be significant in target searches.

The format of the SET CASE subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>CASE</th>
<th>Uppercase</th>
<th>Mixed</th>
<th>Respect</th>
<th>Ignore</th>
</tr>
</thead>
</table>

where:

Uppercase
indicates that the editor is to translate all lowercase letters to uppercase before the letters are entered into the file (whether the lines are entered from the terminal or from the console stack). U is the default value for all filetypes except SCRIPT and MEMO.

Mixed
indicates that the editor is to insert uppercase and lowercase letters in the file in the same way they are entered. No letters are translated.

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:
Mixed Respect (for SCRIPT and MEMO files)
Uppercase Respect (for all other filetypes)

Error 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
SET CMDLINE

Use the CMDLINE option to specify the position of the command line on the screen.

The format of the SET CMDLINE subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>CMDline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>Top</td>
</tr>
<tr>
<td></td>
<td>Bottom</td>
</tr>
</tbody>
</table>

where:

On  
defines the last two lines of the screen as the command input lines.

Top  
defines the top of the screen as the command line (same line as the message line).

Bottom  
defines the last line on the screen as the command input line.

Initial Setting:
CMDLINE ON

Usage Note:
When CMDLINE is set to TOP or BOTTOM, no XEDIT status area is displayed.

Error Messages:
526E OPTION 'CMDLINE' VALID IN DISPLAY MODE ONLY,RC=3
520E INVALID OPERAND : operand,RC=5
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
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.

The format of the SET COLPTR subcommand is:

| [SET] | COLPtr | ON  
|       |        | OFF |

 where:
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.

Error 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
Use the CTLCHAR option to define a control character(s), which specifies that part of a reserved line is to be displayed with or without the following features:

- highlighting
- 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.

The format of the SET CTLCHAR subcommand is:

```
[SET] CTLchar char Escape
      OFF
      Protect
      Noprotect [ High | Nohigh | Invisible ]
      OFF
```

**where:**

- **char**
  is any character, 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.

- **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.

- **High**
  specifies that the string following “char” is to be displayed highlighted. This operand must be used with PROTECT or NOPROTECT.

- **Nohigh**
  specifies that the string following “char” is to be displayed not highlighted (normal intensity). This operand must be used with PROTECT or NOPROTECT.

- **Invisible**
  defines a field where the characters are not displayed, for example, a password. This operand must be used with PROTECT or NOPROTECT.

**Initial Setting:**

No control characters are defined. CTLCHAR is set OFF.

**Notes for Macro Writers:**

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
SET CTLCHAR

origin of each line is inserted. For more information on using READ with the TAG option, see the READ subcommand.

Error Messages:
526E OPTION 'CTLCHAR' VALID IN DISPLAY MODE ONLY., RC=3
520E INVALID OPERAND: operand, RC=5
545E MISSING OPERAND(S), RC=5
554E NO STORAGE AVAILABLE., RC=104

Return Codes:
0 Normal
3 Operand is valid only for display terminal
5 Missing or invalid operand
104 No storage available

Examples:
The following subcommands are issued:

SET CTLCHAR ! ESCAPE
Defines ! as a control character escape. 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 escape, the data that follows it is displayed protected and highlighted.

SET CTLCHAR " PROTECT NOHIGH
When " appears in the text, preceded by the control character escape, the data that follows it is displayed protected and not highlighted.

SET RESERVED 3 NOH This is an !%example!" of selective highlighting.
When the screen is displayed, the word “example” is highlighted on line 3.
**SET CURLINE**

Use the **CURLINE** option to define the nth line of the screen as the current line.

The format of the **SET CURLINE** subcommand is:

```
[SET] CURLine ON n
```

**where:**

- **n**
  - is the screen line number that is to be the current line.

**Initial Setting:**

```
CURLINE ON n
n = middle of the screen
```

**Usage Note:**

Remember that the editor uses the first two lines of the screen; if you want the current line to be the first available screen line, use **SET CURLINE ON 3**.

**Error Messages:**

- 526E OPTION 'CURLINE' VALID IN DISPLAY MODE ONLY,RC=3
- 520E INVALID OPERAND : operand,RC=5
- 521E INVALID LINE NUMBER,RC=5
- 543E INVALID NUMBER : xxxxxxxx,RC=5
- 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
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.

The format of the SET ESCAPE subcommand is:

```
[ SET ] ESCape  ON  [ char ] OFF
```

*where:*

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
```

(A default escape character is assigned according to filetype; 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 may be specified in hexadecimal if SET HEX ON is in effect.

*Error 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
```
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.

The format of the SET FILLER subcommand is:

```
[SET] FILLer [char]
```

Initial Setting:
The “blank” character is defined as the filler.

Usage Note:
The filler character may be specified in hexadecimal if SET HEX ON is in effect.

Error 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
SET FMODE

Use the FMODE option to change the filemode of the file being edited.

The format of the SET FMODE subcommand is:

```
[SET] FMode fm
```

where:

*fm* is the new filemode. You can specify a filemode letter (A-Z) or a filemode letter and number (0-5). If you specify only a filemode letter, the existing filemode number is used.

Usage Notes:

1. The specified filemode is used the next time a FILE or SAVE is issued. If the file being edited had been written to disk before, that copy of the file remains unchanged.

2. If the disk specified by filemode already contains a file with the same filename and filetype, that file is replaced when a FILE or SAVE is issued; no warning message is issued.

3. If the filemode specified is that of a read-only disk, and an attempt is made to file or save the file, the editor displays an error message.

Error Messages:

- `555E FILE 'fn ft fm' ALREADY IN STORAGE,RC=4`
- `520E INVALID OPERAND : operand,RC=5`
- `545E MISSING OPERAND(S),RC=5`
- `048E INVALID MODE 'mode',RC=24`
- `069E DISK 'mode' NOT ACCESSED,RC=36`

Return Codes:

- 0 Normal
- 4 File already in storage
- 5 Missing or invalid operand
- 6 Subcommand rejected in the profile due to LOAD error
- 24 Invalid filemode
- 36 Corresponding disk not accessed
SET FNAME

Use the FNAME option to change the filename of the file being edited.

The format of the SET FNAME subcommand is:

\[
\text{[SET]} \quad \text{FName} \quad fn
\]

where:

\( fn \)

is the new filename; it can be from one to eight characters long.

Usage Notes:

1. The specified filename is used the next time a FILE or SAVE is issued. If the file being edited had been written to disk before, that copy of the file remains unchanged.

2. If a file already exists with the specified filename and the same filetype and filemode, that file is replaced; no warning message is issued.

Error Messages:

555E FILE 'fn ft fm' ALREADY IN STORAGE,RC=4
520E INVALID OPERAND : operand,RC=5
545E MISSING OPERAND(S),RC=5
062E INVALID CHARACTER IN FILEID 'fn ft fm',RC=20

Return Codes:

0 Normal
4 File already in storage
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
20 Invalid character in the string proposed as filename
**SET FTYPE**

Use the FTYPE option to change the filetype of the file being edited.

The format of the SET FTYPE subcommand is:

\[
\begin{array}{|c|c|}
\hline
\text{[SET]} & \text{FType } ft \\
\hline
\end{array}
\]

*where:*

*ft*

is the new filetype; it can be from one to eight characters long.

**Usage Note:**

The specified filetype is used the next time a FILE or SAVE is issued. If the file being edited had been written to disk before, that copy of the file remains unchanged.

**Error Messages:**

- 555E FILE 'fn ft fm' ALREADY IN STORAGE,RC=4
- 520E INVALID OPERAND : operand,RC=5
- 545E MISSING OPERAND(S),RC=5
- 062E INVALID CHARACTER IN FILEID 'fn ft fm',RC=20

**Return Codes:**

- 0 Normal
- 4 File already in storage
- 5 Missing or invalid operand
- 6 Subcommand rejected in the profile due to LOAD error
- 20 Invalid character in the string proposed as filetype
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.

The format of the SET HEX subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>HEX</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
</table>

where:

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

Error 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
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).

The format of the SET IMAGE subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>IMAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>Canon</td>
</tr>
</tbody>
</table>

where:

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.
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
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:

IMAGE ON (all filetypes except SCRIPT)
IMAGE CANON (SCRIPT files)

Usage Notes:
The following subcommands are affected by the IMAGE setting:

- COMPRESS
- EXPAND
- FIND, FINDUP, NFIND, NFINDUP
- INPUT
• OVERLAY
• REPLACE
• all targets
• typing over a line

Error 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
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.

The format of the SET IMPCMSCP subcommand is:

<table>
<thead>
<tr>
<th>SET</th>
<th>IMPCMSCP</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
</table>

where:

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

Error 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
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.

The format of the SET LINEND subcommand is:

```
[SET] LINEND ON [char] OFF
```

where:

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. The default line end character is a pound sign (#).

OFF
specifies that the editor is not to recognize a line end character, without changing the character currently defined.

Initial Setting:
LINEND ON #

Usage Note:
The line end character may be specified in hexadecimal if SET HEX ON is in effect.

Error 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

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.

The format of the SET LRECL subcommand is:

```
[SET] LRecl n
   *
```

where:

- `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.

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.)

2. If the new logical record length is smaller than the previous one, data may be lost when the file is written to disk. 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:

   \[ \text{ZONE1} \leq \text{COLUMN POINTER} \leq \text{ZONE2} \leq \text{TRUNC} \leq \text{LRECL} \leq \text{WIDTH} \]

3. Refer to the LOAD subcommand for more details on the initial setting of LRECL during profile execution.

Initial Setting:

Based on filetype. See "Appendix A".

Error 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 : xxxxxxxxxx,RC=5`
- `545E MISSING OPERAND(S),RC=5`

Return Codes:

- `0` Normal
- `4` Lrec1 must be lower than 65535 for recfm V
- `5` Missing or invalid operand
- `6` Subcommand rejected in the profile due to LOAD error
SET MACRO

Use the MACRO option to control the order in which the editor searches for subcommands and macros.

The format of the SET MACRO subcommand is:

```
SET MACRO ON
OFF
```

where:

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. (See also SET SYNONYM.)

2. The subcommand name SET is required with this option (to avoid conflict with the MACRO subcommand).

Error 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
SET MASK

Whenever a new line is inserted in a file (whether by the ADD subcommand, the A
prefix subcommand, or the INPUT subcommand), it is pre-filled 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.

The format of the SET MASK subcommand is:

```
[SET] MASK Define
     Immed[text]
     Modify
```

where:

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. You can use the QUERY MASK subcommand to display the current mask in the
message line, but you cannot change it.

2. 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.
Pressing only the ERASE EOF key does not clear the mask. (This prevents you
from clearing the mask if you accidentally press ERASE EOF).

Note for Macro Writers:
The TRANSFER MASK subcommand places the current mask in the console stack.

Error Messages:

526E OPTION 'MASK' VALID IN DISPLAY MODE ONLY,RC=3
520E INVALID OPERAND : operand,RC=5
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

Examples:

A mask is analogous to a pre-printed form that has the same information on every
line, and you type in the variable information.

Figure 3-9 is an example of using SET MASK to define the “comments” area in a PLI
program.
1. Enter the following subcommand:

```plaintext
===> SET MASK DEFINE
```

2. The scale appears in the command line:

```plaintext
===> ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*
     ....8....*....9....*....10....*....11....*....12....*....13... XEDIT 1 FILE
```

3. Type over the scale to define the mask:

```plaintext
===> /*
     XEDIT 1 FILE
```

4. If the INPUT subcommand is entered, each line in the input zone contains the mask:

```plaintext
PROGRAM PLI   AI F 80 TRUNC=72 SIZE=12 LINE=0 COLUMN=2
INPUT MODE:

* * * TOP OF FILE * * *
.1.*.1.*.2.*.3.*.4.*.5.*.6.*.7.*.8.*

===> * * * INPUT ZONE * * *
```

Figure 3-9. Using the SET MASK Subcommand
**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
```

where:

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**.

**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, **TRANSFER LASTMSG** can be used to get the last message generated.

2. If multiple messages need to be displayed, they are passed to CMS, the screen is cleared, and the messages are displayed. Press the **CLEAR** key to re-display the file. In this case, the **SET CMSTYPE** command setting determines whether or not the messages are displayed.

**Error 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
SET NONDISP

Use the NONDISP option to define a character to be used in place of non-displayable characters.

The format of the SET NONDISP subcommand is:

\[
\text{[SET]} \ \text{NONDisp} \ [\text{char}]
\]

where:

\[
\text{char}
\]

specifies a character to be used. If not specified, a blank will be used.

Usage Notes:
1. The non-displayable character may be specified in hexadecimal if SET HEX ON is in effect.
2. Setting non-displayable characters 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 non-displayable 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.

Initial Setting:
The initial setting is a blank.

Error Messages:
526E OPTION 'NONDISP' VALID IN 3270 MODE ONLY.,RC=3
520E INVALID OPERAND : operand,RC=5
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
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').

The format of the SET NULLS subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>NULLS ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>

where:

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.

Initial Setting:
NULLS OFF

Usage Notes:
1. You can use the PA2 key instead of issuing SET NULLS ON if you wish to use the insert key for only one line. The PA2 key sets nulls on in the line that contains the cursor. If you move the cursor to another line and wish to use insert mode, you must press the PA2 key again.

2. If either the WIDTH option on the XEDIT command or a SET VERIFY subcommand is set to a value less than the screen width, you can use the insert key to insert characters with SET NULLS OFF.

Error Messages:
526E OPTION 'NULLS' VALID IN DISPLAY MODE ONLY.,RC=3
520E INVALID OPERAND : operand,RC=5
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
SET NUMBER

Use the NUMBER option to specify whether or not file line numbers are to be displayed in the prefix area.

The format of the SET NUMBER subcommand is:

```
<table>
<thead>
<tr>
<th>[SET]</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>
```

where:

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.

Initial Setting:
NUMBER OFF

Error Messages:
526E OPTION 'NUMBER' VALID IN DISPLAY MODE ONLY.,RC=3
520E INVALID OPERAND : operand,RC=5
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
SET PACK

Use the PACK option to specify whether or not the editor is to pack the file when it is written to disk.

The format of the SET PACK subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>PACK</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
</table>

where:

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.

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 that is on disk in packed format, the editor automatically issues a SET PACK ON subcommand. Thus, when the file is filed or saved, it will be written back to disk in packed format.

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.

Responses:

The editor displays the record format as:

FP (fixed packed)

or

VP (variable packed)

Error 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
SET PFn

Use the PFn 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.

The format of the SET PFn subcommand is:

```
[SET] PFn [string]
  TABKEY
  COPYKEY
```

where:

n
specifies a PF key number. If SET PFn is issued (without an additional operand), the meaning associated with that PF key is removed.

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.

TABKEY
When the corresponding PF key is pressed, the cursor is moved into the next tab column, as defined by the SET TABS subcommand. This is the initial setting for the PF4 key.

COPYKEY
When the corresponding PF key is pressed, the exact content of the current physical screen is copied into the printer spool.

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. To get the same effect as a CP SET PF DELAY, use:
   ```
   SET PFn CMSG...
   ```

3. PF keys may be used in input mode. (See the “Usage Notes” section of the INPUT subcommand.)

4. When you use a PF key set to COPYKEY, you should close the printer at the end of the editing session.

5. When a PF key is pressed, its definition is executed before any commands entered in the command line. TABKEY and COPYKEY are executed immediately, and any commands typed in the prefix area or the command line are not executed.

Initial Setting:
```
SET PF01 HELP MENU
SET PF02 SOS LINEADD
SET PF03 QUIT
SET PF04 TABKEY
SET PF05 SCHANCE 6
SET PF06 ?
SET PF07 BACKWARD
```
SET PFn

SET PF08 FORWARD
SET PF09 =
SET PF10 SPLIT CURSOR
SET PF11 JOIN CURSOR
SET PF12 CURSOR COLUMN

In addition, the PA2 key is permanently assigned to SOS NULLS ON.

Note: 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.

Error Messages:
520E INVALID OPERAND : operand,RC=5
525E INVALID PFKEY NUMBER,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
SET POINT

Use the POINT option to define (or redefine) a symbolic name for the current line. You can use this name to refer to the line in subsequent target operands of XEDIT subcommands.

The format of the SET POINT subcommand is:

```plaintext
[SET] Point .symbol [OFF]
```

where:

- **.symbol** is a symbolic name for the current line. The name must begin with a period and be followed by from one to eight alphameric 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:

   ```plaintext
   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:

   ```plaintext
   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. After a symbolic name is defined for a line, the name does not appear in the prefix area, as the line number does. You must keep track of symbolic names.

4. 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.

5. You can use the QUERY POINT subcommand to display the symbolic name of the current line. You can use QUERY POINT * to display all symbolic names that have been defined.

Notes for Macro Writers:

The TRANSFER POINT subcommand places the symbolic name of the current line in the console stack.

Error Messages:

- 540E NAME ALREADY DEFINED ON LINE 'nn'. ,RC=1
- 539E NAMED LINE NOT FOUND. ,RC=2
- 520E INVALID OPERAND: operand, RC=5
- 541E INVALID NAME,RC=5
- 545E MISSING OPERAND(S),RC=5
- 554E NO STORAGE AVAILABLE.,RC=104
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
104 No storage available
**SET PREFIX**

Use the `PREFIX` option to control the display of the prefix area, or to define a synonym for a prefix subcommand.

The format of the `SET PREFIX` subcommand is:

| SET | PREFIX | ON [Left|Right] | OFF |
|-----|--------|----------------|-----|
|     | PREFIX | Synonym newname oldname |

*where:*

**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 can then be entered in the prefix area.

**OFF**

Removes the prefix area from the screen.

**Synonym**

Is the keyword operand that indicates a synonym is to be defined for a prefix subcommand.

**newname**

Is a synonym to be assigned to a prefix subcommand. The new name must be one alphabetic character.

**oldname**

Is the name of a prefix subcommand for which you are defining a synonym. It must be one alphabetic character.

**Initial Setting:**

`PREFIX ON LEFT`

No prefix synonyms are defined.

**Usage Note:**

Synonyms cannot be defined for the prefix subcommands `SCALE` or `TABL`.

**Error Messages:**

526E OPTION 'PREFIX' VALID IN DISPLAY MODE ONLY.,RC=3

520E INVALID OPERAND : operand,RC=5

545E MISSING OPERAND(S),RC=5

548E INVALID SYNONYM OPERAND : operand,RC=5

554E NO STORAGE AVAILABLE,RC=104

**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

104 No storage available
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.

The format of the SET RANGE subcommand is:

\[ [SET] \text{ RANGE } \text{target1} \text{ target2} \]

where:

\text{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 \textit{VM/SP: System Product Editor User’s Guide}.

\text{target2}

is the bottom of the range.

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). The BOTTOM subcommand moves the line pointer to the line before target2. Target2 becomes the equivalent of the EOF 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 on disk. 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:

\text{SET RANGE :1 *}

Error Messages:

546E TARGET NOT FOUND.,RC=2
520E INVALID OPERAND : operand,RC=5
528E INVALID RANGE : TARGET2 (LINE \text{nn}) PRECEDES TARGET1 (LINE \text{nn}).,RC=5
545E MISSING OPERAND(S),RC=5

Return Codes:

0 Normal
2 Target not found
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
Use the RECFM option to define the record format for the file.

The format of the SET RECFM subcommand is:

| [SET] | REC FM | F | V | FP | VP |

where:

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 filetype. See “Appendix A”.

Usage Note:
When the record format is FP or VP, a SET PACK ON is automatically executed.

Error Messages:
516E LRECL TOO LARGE FOR V-FORMAT FILE.,RC=4
515E RECFM MUST BE F|V|FP|VP.,RC=5
520E INVALID OPERAND : operand,RC=5
545E MISSING OPERAND(S),RC=5

Return Codes:
0 Normal
4 Lrecl must be lower that 65536 for recfm V
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
**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, which can be highlighted. 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.

The format of the `SET RESERVED` subcommand is:


*where:*

- `n` specifies the line number that is to be reserved.
- `+n` (the default) indicates that the line number is relative to the top of the screen. `-n` indicates that the line number is relative to the bottom of the screen. For example, `-1` indicates the last line on the screen.
- `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.
- `text` specifies the information that is to be displayed in the reserved line.
- `Off` indicates that a line (n) reserved previously is to be returned for use by the editor.

**Usage Note:**
The `QUERY RESERVED` subcommand may be used to display the line numbers of reserved lines.

**Notes for Macro Writers:**
1. The `TRANSFER RESERVED` subcommand places one line, containing the reserved line numbers, in the console stack.
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.
4. When turning a reserved line off, you must specify “n” the same way you defined it (either plus or minus).
5. 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.
6. A `SET RESERVED` subcommand may be issued for the command line, after which no input is accepted from the command line. If `SET CMDLINE` is in
effect, the command line occupies two lines. Issuing a SET RESERVED for the first line reserves both of the lines.

7. For information on defining various attributes for fields within a reserved line, see the SET CTLCHAR subcommand.

Response:
The information specified in the text operand, if any, appears on the nth line of the screen.

Error Messages:
526E OPTION 'RESERVED' VALID IN DISPLAY MODE ONLY.,RC=3
520E INVALID OPERAND : operand,RC=5
533E LINE 'nn' IS NOT RESERVED,RC=4
521E INVALID LINE NUMBER,RC=5
543E INVALID NUMBER : xxxxxxxx,RC=5
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
6 Subcommand rejected in the profile due to LOAD error
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.

The format of the SET SCALE subcommand is:

```
[SET] SCALE ON [n] OFF
```

where:

ON
displays the scale on the screen.

OFF
removes the scale from the screen.

n is a line number on the screen where the scale is to be displayed.

Initial Setting:
SCALE ON n, where n is the line under the current line.

Usage Notes:
1. The scale looks like this:

```
<...| 1...| 2...| 3...| 4T...| 5...| 6...|
. . . .
. . . .
. . . .
.column pointer . . .
.left zone . . . .
.right zone
```

2. If the current line occupies more than one screen line, the scale is not displayed for that line. The scale is re-displayed when the line pointer moves to a file line that occupies only one screen line.

Error Messages:
526E OPTION 'SCALE' VALID IN DISPLAY MODE ONLY.,RC=3
520E INVALID OPERAND : operand,RC=5
521E INVALID LINE NUMBER,RC=5
543E INVALID NUMBER : xxxxxxxx,RC=5
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
SET SCREEN

Use the SCREEN option to divide the physical 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, status area, and message line.

The format of the SET SCREEN subcommand is:

```
[SET] Screen n
      Size n1 [n2]...
```

where:

- `n` specifies the number of logical screens that the physical screen is to be divided into. The logical screens must be at least five lines long.
- `n1 n2...` specifies the number of lines for each logical screen, thus making it possible to have logical screens of different sizes.

Initial Setting:

```
SCREEN SIZE n1, where n1 is the physical screen size.
```

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 physical screen.

2. If the number of logical screens is decreased due to a SET SCREEN subcommand, files will still be displayed, beginning with the top of the physical screen, as long as logical screens are available. Those files for which logical screens are no longer available are removed from the display.

3. When adding a logical screen, you can control which file is to be displayed. For example, if two files are already displayed, the following subcommands will add a third screen, which will display the file "A LISTING".

   `SCR 3#SOS TABCMDF # XEDIT A LISTING`

4. The SET SCREEN subcommand retains the CURLINE, SCALE, TABLINE, and CMDLINE locations on the screens provided that these settings fall within the new screen size. Otherwise, the default settings are used.

Responses:

The status area of each logical screen contains the number of files being edited.

If the number of screens specified (n) is too large for the physical screen, the editor issues the message:

```
534E TOO MANY LOGICAL SCREENS DEFINED.
```
Error Messages:
536E NUMBER OF LINES EXCEEDS PHYSICAL SCREEN SIZE., RC=1
526E OPTION 'SCREEN' VALID IN DISPLAY MODE ONLY., RC=3
520E INVALID OPERAND : operand, RC=5
534E TOO MANY LOGICAL SCREENS DEFINED, RC=4
537E EACH LOGICAL SCREEN MUST CONTAIN AT LEAST 5 LINES, RC=4
543E INVALID NUMBER : xхххххххххххххх, RC=5
545E MISSING OPERAND(S), RC=5

Return Codes:
0 Normal
1 The total number of lines for the multiple logical screens exceeds the physical screen size
3 Operand is valid only for display terminal
4 Each logical screen must contain at least 5 lines
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
SET SERIAL

Use the SERIAL option to control file serialization.

The format of the SET SERIAL subcommand is:

```
[SET] SERIAL ON [ incrno ] [ startno ]

ALL [ incrno ] [ startno ]

string [ incrno ] [ startno ]

OFF
```

where:

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 filename 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.

Usage Notes:
1. Only files with a fixed record format can be serialized.
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 *
```

Initial Setting:
Based on filetype. See "Appendix A".
Error Messages:
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,RC=5
545E MISSING OPERAND(S),RC=5
558E WRONG FILE FORMAT FOR SERIALIZATION,RC=5
560E NOT ENOUGH ROOM FOR SERIALIZATION BETWEEN TRUNC AND LRECL

Return Codes:
0  Normal
5  Missing or invalid operand
6  Subcommand rejected in the profile due to LOAD error


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.

The format of the SET SPAN subcommand is:

| [SET] | SPAN | ON [Blank | n] | Noblank | * | OFF |

where:

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
1
The house
was near
right zone
72
```

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 (filetype TEXT, zones 1 72) or a PL/I source program (filetype PLI,
zones 2-72) should use SET SPAN ON NOBLANK, since no implicit blanks are assumed to separate lines.

For example:

left zone
  1
right zone
  72

THE HOUSE';

The target /THE LITTLE HOUSE/ will be found.

Error Messages:
520E INVALID OPERAND : operand,RC=5
543E INVALID NUMBER : xxxxxxxx,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
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 target search, FIND, FINDUP, NFIND, or NFINDUP is not found.

The format of the SET STAY subcommand is:

| [SET] | STAY | ON  
|       |      | OFF |

where:

ON
specifies that the line pointer is *not to move* if a search is unsuccessful. The current line stays the same as before the search occurred.

OFF
specifies that 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 becomes the new current line if the search was in a backward direction.

Initial Setting:
STAY OFF

Usage Notes:
1. SET STAY applies to target searches and FIND family searches issued to the end of file (or range).
2. SET STAY applies also to the CHANGE subcommand. With SET STAY OFF, the last line examined becomes the new current line; with SET STAY ON, the line pointer does not move when the CHANGE is executed.

Error 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
**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.

The format of the `SET STREAM` subcommand is:

```
[SET] STREAM ON OFF
```

where:

**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
```

**Error 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
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 (except prefix subcommands) or macro and, optionally, to define an abbreviation for the synonym. (You must use the SET PREFIX subcommand to define a synonym for a prefix subcommand.)

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. For example, if the synonym represents an EDGAR subcommand to be used in EDGAR compatibility mode, you can enter the synonym with its operands in the order normally expected by EDGAR. The operands are automatically rearranged into the order expected by XEDIT.

The format of the SET SYNONYM subcommand is:

```
[SET]  SYNonym  ON
       OFF

SYNonym  newname  [n]  oldname

SYNonym  newname  [n[format1...formatn]]  oldname  [&1...&n]
```

where:

ON

specifies that the editor is to look for synonyms.

OFF

specifies that the editor is not to look for synonyms.

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 represent the first operand in the synonym operand list, &2
represent 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:

<table>
<thead>
<tr>
<th>SET SYNONYM</th>
<th>ALTERNATIVE</th>
<th>2</th>
<th>ALTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET SYNONYM</td>
<td>CAPPEND</td>
<td>2</td>
<td>CAPPEND</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>HEXTYPE</td>
<td>4</td>
<td>HEXTYPE</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>JOIN</td>
<td>1</td>
<td>JOIN</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>MODIFY</td>
<td>3</td>
<td>MODIFY</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>POWERINP</td>
<td>3</td>
<td>POWERINP</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>QUIT</td>
<td>4</td>
<td>COMMAND PQUIT</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>QUIT</td>
<td>2</td>
<td>COMMAND QUIT</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>SPLIT</td>
<td>2</td>
<td>SPLIT</td>
</tr>
<tr>
<td>SET SYNONYM</td>
<td>STATUS</td>
<td>4</td>
<td>STATUS</td>
</tr>
</tbody>
</table>

**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.

**Error Messages:**

520E INVALID OPERAND : operand,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
554E NO STORAGE AVAILABLE.,RC=104

**Return Codes:**

0 Normal
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
104 No storage available
Examples:

Simple form:
1. SYN DOWN 1 UP
2. SYN DSPF QUERY PF

Complex form:
1. SYN PUTFILE 3 & & & PUT &4 &1 &2 &3
   This example allows you to enter the subcommand the way you normally would enter the PUT subcommand under EDGAR:
   PUT fn ft fm n
   The editor rearranges the subcommand as follows:
   PUT target fn ft fm
   Target is limited to an integer value (n) or .xxxx as specified by EDGAR syntax.
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.

The format of the `SET TABLINE` subcommand is:

| SET | TABLine ON [n] OFF |

*where:*

**ON**
- displays the tab line.

**n**
- specifies the line in which the tab line is to be displayed.

**OFF**
- removes the tab line from the screen.

**Initial Setting:**

`TABLINE OFF n`, where `n` is immediately above the command line.

**Usage Note:**

`TABLINE` can be set on the same line as the scale line (see `SET SCALE`).

**Response:**

The specified line contains a "T" in every tab column.

**For example:**

```
T T T T T T T T
```

**Error Messages:**

- `526E OPTION 'TABLINE' VALID IN DISPLAY MODE ONLY,RC=3`
- `520E INVALID OPERAND : operand,RC=5`
- `521E INVALID LINE NUMBER,RC=5`
- `543E INVALID NUMBER : xxxxxxxx,RC=5`
- `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
SET TABS

Use the TABS option to define the logical tab stops for a file.

The format of the SET TABS subcommand is:

```
[SET]      TABS n1[n2...n28]
```

where:

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.

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 filetype 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.

Initial Setting:

Based on filetype. See "Appendix A".

Error Messages:

520E INVALID OPERAND : operand,RC=5
545E MISSING OPERAND(S),RC=5
575E INVALID TABS COLUMNS DEFINED.,RC=5

Return Codes:

0 Normal
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
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.)

The format of the SET TERMINAL subcommand is:

| [SET] | TERMINAL Typewriter Display |

where:

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.

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 a SET TERMINAL DISPLAY subcommand after you reconnect and issue BEGIN. Otherwise the editor resumes editing your file in line mode.

Error Messages:
526E OPTION 'TERMINAL' VALID IN DISPLAY MODE ONLY., RC=3
520E INVALID OPERAND : operand, RC=5
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
SET TEXT

Use the TEXT option to inform the editor if TEXT keys are available on the terminal.

The format of the SET TEXT subcommand is:

<table>
<thead>
<tr>
<th>SET</th>
<th>TEXT</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
</table>

where:

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:
According to the CP TERMINAL TEXT setting when the editor is invoked.

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. In CP mode, you can issue a TERMINAL TEXT ON command, which will be recognized when the editor is invoked.
   b. If the CP TERMINAL command has not been issued in CP mode, you must issue the editor SET TEXT subcommand.

   If instead, you issue a TERMINAL command directly to CP while in edit mode (via the CP or CMS subcommand or in subset mode), the editor will not recognize the command and will not perform the necessary conversions.

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.

Error Messages:
524W NONDISP CHARACTER RESET TO BLANK.
526E OPTION 'TEXT' VALID IN DISPLAY MODE ONLY, RC=3
520E INVALID OPERAND: operand, RC=5
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
SET TOEEOF

Use the TOEEOF option to control the display of the following null lines:

- TOP OF FILE
- END OF FILE
- TOP OF RANGE
- END OF RANGE

The format of the TOEEOF subcommand is:

```
[SET] TOEEOF ON
    OFF
```

*where:*

- **ON** specifies that the notices listed above are displayed.
- **OFF** specifies that the notices listed above are not displayed.

**Initial Setting:**

TOEEOF ON

**Error 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
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.

The format of the SET TRUNC subcommand is:

```
[SET] TRunc n *
```

where:

- `n` specifies the column at which truncation is to occur. If you specify an asterisk (*), the truncation column is set to the record length for the filetype.

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.

Initial Setting:
Based on filetype. See “Appendix A”.

Error Messages:
- 009E COLUMN 'nn' EXCEEDS RECORD LENGTH (nn),RC=5
- 520E INVALID OPERAND : operand,RC=5
- 543E INVALID NUMBER: xxxxxxxx,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
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.

The format of the SET VARBLANK subcommand is:

```
[SET] VARblank [ON | OFF]
```

where:

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.

Error 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
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.

The format of the SET VERIFY subcommand is:

```
[SET] Verify[ON ] [[H] startcol endcol] ...
```

where:

- **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.)

- **H**
  - 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 filetype. See “Appendix A”.

Usage Notes:

1. Up to 14 pairs of columns may be specified. For example:

   \[V\ 1\ 20\ 40\ 50\]

   displays columns 1 through 20, and 40 through 50.

   However, when multiple pairs of columns are specified, the results of some subcommands (for example, SET TABLINE ON) may be unpredictable.

2. An area can be displayed in both EBCDIC and hexadecimal. For example:

   \[V\ 1\ 20\ H\ 120\]

   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.

4. In multiple views of the same column, if changes are made on the screen the right-most change is the effective one.

5. The columns specified in a SET VERIFY subcommand override any RIGHT or LEFT subcommand that was in effect.

6. 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 filetype 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 172. (You could then use a RIGHT subcommand to view the columns of data extending past column 72.)
Error Messages:
009E COLUMN 'nn' EXCEEDS RECORD LENGTH (nn)..,RC=5
520E INVALID OPERAND : operand,RC=5
545E MISSING OPERAND(S),RC=5
575E INVALID VERIFY COLUMNS DEFINED.,RC=5
576E TOTAL VERIFY WIDTH EXCEEDS SCREEN SIZE (nn)..,RC=5

Return Codes:

0 Normal
5 Missing or invalid operand
6 Subcommand rejected in the profile due to LOAD error
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, or a target search.

The format of the SET WRAP subcommand is:

<table>
<thead>
<tr>
<th>[SET]</th>
<th>WRap</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
</table>

where:

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

Error 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
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.

The format of the SET ZONE subcommand is:

<table>
<thead>
<tr>
<th></th>
<th>Zone</th>
<th>zone1</th>
<th>zone2</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SET] Zone</td>
<td>zone1</td>
<td>zone2</td>
<td>*</td>
</tr>
</tbody>
</table>

where:

- **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.)

**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.
- If the current setting of the column pointer is greater than zone2, it moves to the column defined by zone2.

**Initial Setting:**

The ZONE is set from column 1 up to the truncation column.

**Error Messages:**

- 520E INVALID OPERAND : operand,RC=5
- 543E INVALID NUMBER : xxxxxxxx,RC=5
- 545E MISSING OPERAND(S),RC=5
- 575E INVALID ZONE COLUMNS DEFINED,,RC=5
Return Codes:
0  Normal
5  Missing or invalid operand
6  Subcommand rejected in the profile due to LOAD error
SET =

Use the = option to insert the specified string into the equal (=) buffer. (See the = subcommand.)

The format of the SET = subcommand is:

| SET = string |

Error Messages:
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
SHIFT

Use the SHIFT subcommand to move data to the right or to the left. Data loss is possible.

The format of the SHIFT subcommand is:

<table>
<thead>
<tr>
<th>SHIFT</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cols [target]</td>
<td></td>
</tr>
</tbody>
</table>

where:

Left
shifts data to the left. Data shifted to the left past column one 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 is lost. 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 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. The line pointer is moved to the last line shifted. The column pointer is not affected by SHIFT.

Error Messages:
- 546E TARGET NOT FOUND,,RC=2
- 504E nn LINE(S) TRUNCATED,,RC=3
- 585E NO LINE(S) CHANGED,,RC=4
- 520E INVALID OPERAND : operand,,RC=5
- 543E INVALID NUMBER,,RC=5
- 545E MISSING OPERAND(S),,,RC=5

Return Codes:
- 0 Normal
- 1 TOP or EOF reached during execution
- 2 Target line not found
- 3 Lines truncated
- 4 No lines changed
- 5 Invalid operands
- 6 Subcommand rejected in the profile due to LOAD error
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.

The format of the SORT macro is:

```
SORT  target  [AD]  col1  col2  [col1  col2]...
```

where:

- **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.

- **A** 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 255.

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. If the SORT macro is used while in update mode, the changes made by SORT will not be reflected in the update file. If the only changes to the file are those made by SORT, no update file is created.

Error Messages:

- 546E TARGET NOT FOUND,RC=2
- 520E INVALID OPERAND: operand,RC=5
- 545E MISSING OPERAND(S),RC=5
- 588E PREFIX SUBCOMMAND WAITING...,RC=8
- 009E COLUMN EXCEEDS RECORD LENGTH,RC=24
- 053E INVALID SORT FIELD PAIR DEFINED,RC=24
- 063E NO SORT LIST GIVEN,RC=40
- 554E NO STORAGE AVAILABLE,RC=104

Return Codes:

- 0  Normal
- 1  TOF or EOF reached during execution
- 2  Target line not found
- 5  Missing or invalid operand
- 6  Subcommand rejected in the profile due to LOAD error
- 8  Prefix area contains pending or incomplete subcommand
- 24 Invalid columns defined
- 40 No list given
- 104 No storage available
Examples:

```
SORT * 17 24 8 16
```

sorts a CMS EXEC file by filetype (columns 17 through 24) and, within each filetype, by filename (columns 8 through 16).
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.

The format of the SOS subcommand is:

<table>
<thead>
<tr>
<th>SOS</th>
<th>option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options:</td>
<td></td>
</tr>
<tr>
<td>ALarm</td>
<td>Nulls OFF</td>
</tr>
<tr>
<td>BField n</td>
<td>PFn</td>
</tr>
<tr>
<td>CLEAR</td>
<td>POP</td>
</tr>
<tr>
<td>Down n</td>
<td>Push</td>
</tr>
<tr>
<td>Left n</td>
<td>Right n</td>
</tr>
<tr>
<td>LINEAdd</td>
<td>TABB n</td>
</tr>
<tr>
<td>LINEDel</td>
<td>TABCmd</td>
</tr>
<tr>
<td>NField n</td>
<td>TABCMD</td>
</tr>
<tr>
<td>NLine n</td>
<td>TABCMDF</td>
</tr>
<tr>
<td>Nulls</td>
<td>TABF n</td>
</tr>
<tr>
<td>Nulls On</td>
<td>Up n</td>
</tr>
</tbody>
</table>

where:

- **ALarm**
  - Ring the terminal alarm the next time the display is refreshed.

- **BField n**
  - Reposition the cursor as if the back field key had been depressed n times. The default is one (1).

- **CLEAR**
  - Clears the CMS console stack and forces a display refresh.

- **Down n**
  - Move the cursor down n lines on the virtual screen. The default is one (1).

- **Left n**
  - Move the cursor n places to the left. The default is one (1).

- **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.

- **NField n**
  - Move the cursor to the next 3270 field n times. The default is one (1).

- **NLine n**
  - Depress the "new line" key n times. The default is one (1).

- **Nulls**
  - Reverse the current setting of the "nulls" option for the 3270 field in which the cursor is located.

- **Nulls On**
  - Change the trailing blanks to "nulls" for the 3270 field in which the cursor is located. (The PA2 key is permanently assigned to SET NULLS ON.)

- **Nulls OFF**
  - Change the trailing "nulls" to blanks for the 3270 field in which the cursor is located.
PFn
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.)

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.

Right n
Move the cursor n positions to the right on the virtual screen. The default is one (1).

TABB n
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
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
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
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.

Up n
Move the cursor up n lines. The default value for n is one (1).

Usage Note:
Any operand that is not recognized is ignored.

Error Messages:
561E CURSOR IS NOT ON A VALID DATA FIELD.,RC=3
545E MISSING OPERAND(S),RC=5

Return Codes:
0 Normal
3 Cursor is not on a valid data field
5 Missing operands
6 Subcommand rejected in the profile due to LOAD error
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.
The format of the SPLIT macro is:

\[
\text{SPLIT} \begin{bmatrix}
\text{Column} \\
\text{CURSOR}
\end{bmatrix}
\begin{bmatrix}
\text{colno} \\
\text{Before} /\text{string/} \\
\text{After}
\end{bmatrix} \ldots
\]

where:

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 line pointer and column pointer remain 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. The PF10 key is initially set to SPLIT CURSOR.

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.
3. The cursor or the column number or string specified must fall within the current zones.

Responses:
The last line added due to a SPLIT becomes the new current line.
If a specified string is not found, an error message is issued and the current line remains the same.
Error Messages:
585E NO LINE(S) CHANGED,RC=1
586E NOT FOUND,RC=2
561E CURSOR IS NOT ON A VALID DATA FIELD,RC=3
526E OPTION 'CURSOR' VALID IN DISPLAY MODE ONLY, RC=3
557E NO MORE STORAGE TO INSERT LINE,RC=4
575E INVALID SPLIT COLUMNS DEFINED,RC=5
588E PREFIX SUBCOMMAND WAITING,RC=8

Return Codes:
0 Normal
1 No change (SPLIT issued at TOF or EOF)
2 Not found
3 Cursor is not on a valid data field
4 No more storage
5 Invalid operands
6 Subcommand rejected in the PROFILE due to LOAD error
8 Prefix area contains pending or incomplete subcommand

Examples:
Current Line:
===== Electric eels can discharge bursts of 625 volts.
Note the cursor position above. Press the PF key assigned to SPLIT CURSOR (PF10).
===== 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 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 a result of the split.
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.

The format of the STACK subcommand is:

```
STACK [ target [ startcol [ length ] ] ]
```

where:

**target**
- defines the number of lines to be stacked. Lines are stacked starting with the current line, up to but not including target. 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 publication 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. If you omit the length, the whole line is stacked.

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 256 bytes.
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.

Error Messages:
- 546E TARGET NOT FOUND, RC=2
- 520E INVALID OPERAND: operand, RC=5
- 543E INVALID NUMBER: xxxxxxxx, RC=5

Return Codes:
- 0 Normal
- 1 TOP 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
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 and their current settings.

The format of the STATUS macro is:

```
STATUS [filename]
```

Where:

- `filename` specifies the name of a file that is to contain all the SET subcommand options (except for PF, POINT, RESERVED, and =) and their current settings. The editor assigns a filetype 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 for PF, POINT, RESERVED, and =.
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.

Responses:
7031 FILE 'filename XEDIT A1' CREATED.

Error Messages:
- 520E INVALID OPERAND(S) : operand,RC=5
- 024E FILE 'filename XEDIT A' ALREADY EXISTS,RC=28

Return Codes:
- 0 Normal
- 5 Invalid operand
- 6 Subcommand rejected in the PROFILE due to LOAD error
- 28 Filename specified already exists
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).

The format of the TOP subcommand is:

```
TOP
```

Usage Notes:

1. 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.

2. CLOCATE should also be preceded by TOP if a string to be located might be in column one of the first 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.

Error Messages:

520E INVALID OPERAND(S) : operand, RC=5

Return Codes:

1  Top of file reached
5  Invalid operand
6  Subcommand rejected in the profile due to LOAD error
TRANSFER

The TRANSFER subcommand is used within a macro to access specified editing variables, for example, the current line number, the filename 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.

The format of the TRANSFER subcommand is:

<table>
<thead>
<tr>
<th>TRANSfer</th>
<th>APL</th>
<th>PFn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARBchar</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Autosave</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CASE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMDline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLPtr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLUMNn</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CTLchar [char]</td>
<td>SCReen</td>
</tr>
<tr>
<td></td>
<td>CURLine</td>
<td>Seq8</td>
</tr>
<tr>
<td></td>
<td>CURSor</td>
<td>SERIAL</td>
</tr>
<tr>
<td></td>
<td>EOF</td>
<td>SIDcode</td>
</tr>
<tr>
<td></td>
<td>ESCape</td>
<td>SIZE</td>
</tr>
<tr>
<td></td>
<td>FILLer</td>
<td>SPAN</td>
</tr>
<tr>
<td></td>
<td>FMode</td>
<td>STAY</td>
</tr>
<tr>
<td></td>
<td>FName</td>
<td>STREAM</td>
</tr>
<tr>
<td></td>
<td>FType</td>
<td>SYNONYM [name]</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>TABLine</td>
</tr>
<tr>
<td></td>
<td>IMAGE</td>
<td>TABS</td>
</tr>
<tr>
<td></td>
<td>IMPcmscp</td>
<td>TARGET</td>
</tr>
<tr>
<td></td>
<td>LASTmsg</td>
<td>TERMINal</td>
</tr>
<tr>
<td></td>
<td>LENGth</td>
<td>TEXT</td>
</tr>
<tr>
<td></td>
<td>Line</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td>LINENd</td>
<td>TOFEOF</td>
</tr>
<tr>
<td></td>
<td>LRecI</td>
<td>TRunc</td>
</tr>
<tr>
<td></td>
<td>LScreen</td>
<td>UPDATE</td>
</tr>
<tr>
<td></td>
<td>MACRO</td>
<td>VARblank</td>
</tr>
<tr>
<td></td>
<td>MASK</td>
<td>Verify</td>
</tr>
<tr>
<td></td>
<td>MSGMode</td>
<td>VERShift</td>
</tr>
<tr>
<td></td>
<td>NBFile</td>
<td>Width</td>
</tr>
<tr>
<td></td>
<td>NONDisp</td>
<td>WRap</td>
</tr>
<tr>
<td></td>
<td>NULLs</td>
<td>Zone</td>
</tr>
<tr>
<td></td>
<td>NUMBER</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td>PACK</td>
<td></td>
</tr>
</tbody>
</table>

where:

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, fileid, 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.
TRANSFER

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.

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 attribute2”. If no
attributes were defined for the character, transfers “CTLCHAR char”.

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 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.

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 filemode.

FName
 transfers the eight-character filename.
FType
  transfers the eight-character filetype.
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.
LINEEND
  transfers "OFF" and the line end character as defined by the SET LINEEND subcommand.
TRANSFER
TRANSFER

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 physical screen; the number of lines and number of columns of the physical 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” or “OFF” 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 nl n2...", where nl 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 SIZE subcommand.

SEQ8
transfers "OFF" if the XEDIT command was issued with the NOSEQ8 operand; if not, transfers "ON".

SERIAL
transfers the serial identification, 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.
TRANSFER

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 the verification columns specified in the SET VERIFY subcommand.

VERSift
transfers +n or -n, which is the relative position of the screen over the file, as a result of any LEFT, RIGHT, or SET VERIFY 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 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.

Error Messages:
545E MISSING OPERAND(S),RC=5

Return Codes:
0 Normal
5 Invalid operand specified; no information is stacked
6 Subcommand rejected in the profile due to LOAD error
TYPE

Use the TYPE subcommand to display a specified number of lines, starting with the current line.

The format of the TYPE subcommand is:

```
Type [target|1]
```

where:

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 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 up to 130 characters for each record (the maximum length supported by CMS). If the line is longer, it is displayed truncated.

Response:
The specified lines are displayed.
The line pointer moves to the last line typed.

Error Messages:
S46E TARGET NOT FOUND., RC=2
S20E INVALID OPERAND: operand, RC=5

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
Use the `UP` subcommand to move the line pointer a specified number of lines toward the top of the file.

The format of the `UP` subcommand is:

```
    Up   [n|*|1]
```

where:

- `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.

Error Messages:
- `520E INVALID OPERAND : operand,RC=5`
- `543E INVALID NUMBER : xxxxxxxx,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

Examples:
Figure 3-10 is a before-and-after example of the `UP` subcommand.
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.

A guide informed him later, been eaten by an alligator. Professor Twist could not but smile. "You mean," he said, "a crocodile."

"You mean," he said, "a crocodile."
UPPERCAS

Use the UPPERCAS subcommand to translate all lowercase characters to uppercase, starting at the current line, for a specified number of lines.

The format of the UPPERCAS subcommand is:

```
UPPERcas [target]\n```

where:

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:
When you press the ENTER key, all lowercase letters within the current zones appear in uppercase.

The last line translated becomes the new current line.

Error Messages:
546E TARGET NOT FOUND., RC=2
505E NO LINE(S) CHANGED, RC=2
520E INVALID OPERAND : operand, RC=5

Return Codes:
0 Normal
1 TOP 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

Examples:
===== Elephant tusks can weigh up to 300 pounds.
UPP   (translate the current line to uppercase)
===== ELEPHANT TUSKS CAN WEIGH UP TO 300 POUNDS.
Use the XEDIT subcommand to edit multiple files.

The format of the XEDIT subcommand is:

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

where:

- **fn**
  - is the filename 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 filetype of the above file.

- **fm**
  - is the filemode of the above file. If you omit fm, * is assumed (all of the accessed disks are searched).

- **options**
  - are the same as the options in the CMS command XEDIT. (See "Section 2: The XEDIT Command").

Usage Notes:

1. The XEDIT subcommand allows you to edit independently multiple files in virtual storage.

   Each time the XEDIT subcommand is issued with a fileid, that file is brought into storage and becomes the current one.

   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.

   A QUIT or FILE subcommand issued from a file causes the previous file in the ring to be displayed.

2. When the screen is divided into multiple logical screens (see the SET SCREEN subcommand), entering the XEDIT subcommands with the same fileid from two (or more) logical screens creates two (or more) independent views of the same file.

3. The fileid 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.

4. If this subcommand is issued as "CMS XEDIT...", XEDIT will be called recursively instead (see "Section 2: The XEDIT Command," usage note 3.)

Error Messages:

- 062E INVALID CHARACTER IN FILEID 'fn ft fm',RC=20
- 003E INVALID OPTION 'option',RC=24
- 029E INVALID PARAMETER 'parameter' IN THE OPTION 'option' FIELD,RC=24
- 048E INVALID MODE 'mode',RC=24
- 054E INCOMPLETE FILEID SPECIFIED,RC=24
- 065E 'option' OPTION SPECIFIED TWICE,RC=24
- 066E 'option' AND 'option' ARE CONFLICTING OPTIONS.,RC=24
- 070E INVALID PARAMETER 'parameter',RC=24
- 002E FILE 'fn ft fm' NOT FOUND,RC=28
- 024E FILE 'XEDTEMP CMSUT1 A1' ALREADY EXISTS,RC=28
- 069E DISK 'mode' NOT ACCESSSED,RC=36
- 229E UNSUPPORTED OS DATA SET,RC=80,81,82,83
- 132S FILE 'fn ft fm' TOO LARGE,RC=100
- 590E DATA SET TOO LARGE,RC=88
- 104S ERROR 'nn' READING FILE 'fn ft fm' FROM DISK,RC=100

IBM VM/SP: System Product Editor Command and Macro Reference
Error Messages with UPDATE Options:

007E FILE 'fn ft fm' IS NOT FIXED, 80 CHAR. RECORDS.,RC=32
184W ',' / S' NOT FIRST CARD IN UPDATE FILE -- IGNORED
185W NON NUMERIC CHARACTER IN SEQUENCE FIELD '........',RC=32
186W SEQUENCE NUMBER NOT FOUND; ,RC=32
207W INVALID UPDATE FILE CONTROL CARD.,RC=32
174W SEQUENCE ERROR INTRODUCED IN OUTPUT FILE: '........' TO '........', ,RC=32
179E MISSING OR DUPLICATE 'MACS' CARD IN CONTROL FILE 'fn ft fm'.
183E INVALID aux/ctrl FILE CONTROL CARD.,RC=32
210W INPUT FILE SEQUENCE ERROR '........' TO '........',RC=32
597E UNABLE TO MERGE UPDATES CONTAINING '/ S' CARDS.,RC=32

Return Codes:

0 Normal
6 Subcommand rejected in the profile due to LOAD error
20 Invalid character in filename or filetype
24 Invalid parameters, or options
28 Source file not found (UPDATE MODE) or the specified profile macro does not exist, or file XEDITMP CMSUT1 already exists.
32 Error during updating process
36 Corresponding disk not accessed
88 File is too large and does not fit into storage
100 Error reading the file into storage
& (Ampersand)

Use an ampersand (&) in column one of the command line before any subcommand to cause the subcommand to be redisplayed. Using an & allows you to re-execute the subcommand just by pressing the ENTER key.

The format of the & subcommand is:

```
& [ subcommand ]
```

Usage Notes:
1. A synonym cannot be defined for &.
2. & cannot be used in an XEDIT macro.
Use the = subcommand to re-execute the last subcommand or macro entered, or to execute a specified subcommand and then re-execute the last one entered.

The format of the = subcommand is:

```
= [subcommand]
```

where:

- `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 re-executed.

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 re-executed can have been entered from the command input area on the terminal or from the console stack (via a macro).
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. placed in the console stack, without being changed, by using TRANSFER =
   c. specified by using SET = line, where line 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.
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
```
? (Question Mark)

Use the ? subcommand to display in the command area the last XEDIT subcommand executed, except for an \* or ? subcommand.

The format of the ? subcommand is:

```
?
```

Usage Notes:
1. The subcommand that is displayed as a result of a ? can be re-executed by pressing the ENTER key. You can also modify the command before re-entering 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 key. For example, the editor assigns the ? subcommand to the PF6 key (initial setting).

5. Anything following a ? is ignored except another ?. Multiple ?s can be specified to retrieve previous subcommands. For example, ??? displays the third previous subcommand.

Notes for Macro Writers:
The ? subcommand cannot be used in a macro.
Section 4: Prefix Subcommands

Prefix subcommands are "line" subcommands, which are entered by typing over the five-position prefix area. You can use the prefix subcommands to:

- insert and delete 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

The prefix subcommands are:

A  Add (equivalent to I)
C  Copy
D  Delete
E  Extend
F  Following
I  Insert
M  Move
P  Preceding
"Duplicate
/  Set current line
SCALE Display scale
TABL Display tab line
.xxxxx Assign symbolic name

General Usage Notes:

1. Use the subcommands

   SET PREFIX ON RIGHT

   or

   SET PREFIX ON LEFT

   to display the five-position prefix area (=====).

2. Prefix subcommands 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)
   a=3==       (adds 1 line)

   are valid ways to enter prefix subcommands. You can type multiple prefix subcommands before pressing the ENTER key.

3. If line numbers are displayed in the prefix area (via the SET NUMBER ON subcommand), use prefix subcommands carefully. Only the characters that you type over and that are different from the numbers in the prefix area are interpreted as the prefix subcommand.
For example:
The prefix area contains 012345
You type over "a4": 01a445
The prefix subcommand is "a4".
If you typed "a44": 01a445
or if you typed "a445": 01a445
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.
However, if you leave the cursor in the prefix area, the editor interprets the
changed part of the prefix area up to the cursor.
For example:
The prefix area contains: 012345
You type over "a3" (note
the cursor): 01a345
Because the cursor is left in the prefix area, the prefix subcommand is "a3".
4. The RESET subcommand (entered on the command line) can be used to cancel
any pending prefix subcommands, that is, prefix subcommands that have
caus ed a "COPY/MOVE PENDING" or "BLOCK INCOMPLETE" message to be
displayed in the status area. All prefix subcommands that have been typed
but not yet executed will be removed.
5. When you are editing a file on multiple screens (multiple views of the same
file) prefix subcommands may be specified on all of the views.
6. Prefix subcommands are executed only when the sequence entered is syntacti-
cally correct and coherent. Otherwise, none of them are executed and they
are displayed (highlighted) on the screen with an error message or a status
condition in the status area:
Error Messages:
INVALID PREFIX SUBCOMMAND
PREFIX SUBCOMMAND CONFLICT
Status:
BLOCK INCOMPLETE
COPY/MOVE PENDING
7. Prefix subcommands are executed before any commands executed by pressing
a PF key or before any commands that may be typed in the command line.
8. If you type a prefix subcommand 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. If the prefix subcommand also
caus ed a "pending" status, the adjusted number appears in the prefix area,
replacing the number you typed.
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.

The format of the A prefix subcommand is:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>add one line</td>
</tr>
<tr>
<td>nA</td>
<td>add n lines</td>
</tr>
<tr>
<td>An</td>
<td>add n lines</td>
</tr>
</tbody>
</table>

Usage Note:
The A prefix subcommand is equivalent to the I prefix subcommand.

Responses:
The prefix area on each line that is added is highlighted.
Each line that is added is pre-filled with the current mask (see SET MASK).
The cursor is placed in the first tab column of the first line that was added.

Example:
Refer to the "Examples" section of the D prefix subcommand.
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).

The format of the C prefix subcommand is:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>copy line</td>
</tr>
<tr>
<td>Cn</td>
<td>copy n lines</td>
</tr>
<tr>
<td>nC</td>
<td>copy n lines</td>
</tr>
<tr>
<td>CC</td>
<td>copy block of lines</td>
</tr>
</tbody>
</table>

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.
2. To copy one line, enter the character C in the prefix area of the line.
3. To copy more than one line, enter Cn or nC 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).
The message

COPY/MOVE 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:

BLOCK INCOMPLETE

This “incomplete” status allows you to scroll through the file before completing the block.
D (Delete)

Use the D prefix subcommand to delete one line, a specified number of lines, or a block of lines.

The format of the D prefix subcommand is:

<table>
<thead>
<tr>
<th>D</th>
<th>Delete one line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dn</td>
<td>Delete n lines</td>
</tr>
<tr>
<td>nD</td>
<td>Delete n lines</td>
</tr>
<tr>
<td>DD</td>
<td>Delete block of lines</td>
</tr>
</tbody>
</table>

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.

Responses:
When a DD has been entered on only one line of a block of lines to be deleted, the following message is displayed in the status area:

BLOCK INCOMPLETE

This “incomplete” status allows you to scroll through the file before completing the block.

The cursor is placed on the line following the last line deleted.

Examples:
Figure 4-1 is a before-and-after example of the A and D prefix subcommands.
ANIMALS FACTS A1 F 80 TRUNC=80 SIZE=14 LINE=9 COLUMN=1

===== ** TOP OF FILE ****
D==== THE HIPPOPOTAMUS IS DISTANTLY RELATED TO THE PIG.
D==== ELEPHANT TUSKS CAN WEIGH MORE THAN 300 POUNDS.
D==== LAND CRABS FOUND IN CUBA CAN RUN FASTER THAN A DEER.
D==== ELECTRIC EELS CAN DISCHARGE BURSTS OF 625 VOLTS,
D==== 40 TIMES A SECOND.
D==== THE ANCIENT ROMANS AND GREEKS BELIEVED THAT BEDBUGS HAD MEDICINAL
D==== PROPERTIES WHEN TAKEN IN A DRAFT OF WATER OR WINE.
D==== STURGEON IS THE LARGEST FRESHWATER FISH AND CAN WEIGH 2250 POUNDS.
E==== SNakes lay EGGS with NONBRITTLE SHELLS.
E==== THE PLATYPUS HAS A DUCK BILL, OTTER FUR, WEBBED FEET, LAYS
E==== EGGS, AND EATS ITS OWN WEIGHT IN WORMS EVERY DAY.
==== END OF FILE ****

>>> XEDIT 1 FILE

ANIMALS FACTS A1 F 80 TRUNC=80 SIZE=13 LINE=9 COLUMN=1

===== ** TOP OF FILE ****
E==== ELEPHANT TUSKS CAN WEIGH MORE THAN 300 POUNDS.
E==== LAND CRABS FOUND IN CUBA CAN RUN FASTER THAN A DEER.
E==== ELECTRIC EELS CAN DISCHARGE BURSTS OF 625 VOLTS,
E==== 40 TIMES A SECOND.
E====
E====
E==== THE ANCIENT ROMANS AND GREEKS BELIEVED THAT BEDBUGS HAD MEDICINAL
E==== PROPERTIES WHEN TAKEN IN A DRAFT OF WATER OR WINE.
E==== ALL OSTRICHES ARE POLYGAMOUS.
E==== SNakes lay EGGS with NONBRITTLE SHELLS.
E==== THE PLATYPUS HAS A DUCK BILL, OTTER FUR, WEBBED FEET, LAYS
E==== EGGS, AND EATS ITS OWN WEIGHT IN WORMS EVERY DAY.
E==== END OF FILE ****

>>> XEDIT 1 FILE

Figure 4-1. Prefix Subcommands A and D - Before and After
E (Extend)

Use the E prefix subcommand to extend a logical line by one more physical line on the screen.

The format of the E prefix subcommand is:

```
 E
```

Usage Notes:
1. After an E prefix subcommand is entered, the entire logical line, which now appears on two physical 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.
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.

The format of the F prefix subcommand is:

```
F
```

Response:
The message

```
COPY/MOVE 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.

Example:
See Figure 4-2 (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.

The format of the I prefix subcommand is:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Insert one line</td>
</tr>
<tr>
<td>nI</td>
<td>Insert n lines</td>
</tr>
<tr>
<td>In</td>
<td>Insert n lines</td>
</tr>
</tbody>
</table>

**Usage Note:**
The I prefix subcommand is identical to the A prefix subcommand.

**Responses:**
The prefix area of each line that is inserted is highlighted. Each line that is inserted is pre-filled with the current mask (see the SET MASK subcommand). The cursor is placed on the first line that was inserted.
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.

The format of the M prefix subcommand is:

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>move one line</td>
</tr>
<tr>
<td>Mn</td>
<td>move n lines</td>
</tr>
<tr>
<td>nM</td>
<td>move n lines</td>
</tr>
<tr>
<td>MM</td>
<td>move block of lines</td>
</tr>
</tbody>
</table>

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.
2. To move one line, enter the character M in the prefix area of the line.
3. To move more than one line, enter Mn or nM 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 screens.

Responses:
The message COPY/MOVE 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 message is displayed in the status area:

BLOCK INCOMPLETE

This “incomplete” 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.

Examples:
Figure 4-2 is a before-and-after example of the M and F prefix subcommands.
ANIMALS FACTS  A1 V 132 TRUNC=132 SIZE=22 LINE=10 COLUMN=1

====== 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.
====== 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 LIZARD OF CENTRAL AMERICA CALLED THE BASILISK CAN RUN
====== ACROSS WATER.
====== OCTOPI HAVE LARGE BRAINS AND SHOW CONSIDERABLE CAPACITY FOR
====== LEARNING.
====== THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
====== A FISH CALLED THE NORTHERN SEA ROBIN MAKES NOISES LIKE A WET
====== FINGER DREW ACROSS AN INFLATED BALLOON.
====== STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS.

XEDIT FILE

Figure 4-2. Prefix Subcommands M and F - Before and After

Section 4: Prefix Subcommands  4 - 11
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.

The format of the P prefix subcommand is:

```
P
```

Response:
The message

```
COPY/MOVE 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.
SCALE (Display Scale)

Use the SCALE prefix subcommand to display the scale on this line.
The format of the SCALE prefix subcommand is:

```
SCALE
```

Usage Note:
The SCALE prefix subcommand has the same effect as the subcommand:

```
SET SCALE ON n
```
**TABL (Display Tab Line)**

Use the `TABL` prefix subcommand to display a "T" in every tab column, according to the current tab settings, on the corresponding screen line.

The format of the `TABL` prefix subcommand is:

```
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 T T
```
/ (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.

The format of the / prefix subcommand is:

```
/ [n]  or  [n]/
```

where:

- **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. The / prefix subcommand is the only one that is executed even when the status area indicates "BLOCK INCOMPLETE" or "COPY/MOVE PENDING". This allows you to scroll through the file while defining blocks of lines to be deleted, moved, copied, or duplicated.
" (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.

The format of the " prefix subcommand is:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>duplicate one line</td>
</tr>
<tr>
<td>&quot;n or n&quot;</td>
<td>duplicate line n times</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>duplicate block of lines</td>
</tr>
<tr>
<td>&quot;&quot;n or n&quot;&quot;</td>
<td>duplicate block n times</td>
</tr>
</tbody>
</table>

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 message is displayed in the status area:

    BLOCK INCOMPLETE

This “incomplete” status allows you to scroll through the file before completing the block.
Use the .xxxx prefix subcommand to assign a symbolic name to this line. You can use this name to refer to the line in subsequent target operands of XEDIT subcommands.

The format of the .xxxx prefix subcommand is:

```
.xxx
```

where:

.xxx

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).
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 currently defined. The subcommand QUERY POINT can be used to display the name of the current line.
### Appendix A: Filetype Defaults

<table>
<thead>
<tr>
<th>Filetype</th>
<th>SERIAL</th>
<th>TRUNC</th>
<th>LRECL</th>
<th>VERIFY</th>
<th>RECFM</th>
<th>FILLER</th>
<th>ESCAPE</th>
<th>CASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM3705</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>71</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>AMSERV</td>
<td>ON</td>
<td>72</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>BASDATA</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>BASIC</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>CNTRL</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>COBOL</td>
<td>ON</td>
<td>72</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>COPY</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>71</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>DIRECT</td>
<td>ON</td>
<td>72</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>EXEC</td>
<td>OFF</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>ON</td>
<td>72</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>FREEFORT</td>
<td>OFF</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>JOB</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>F</td>
<td>blank</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>LISTING</td>
<td>OFF</td>
<td>121</td>
<td>121</td>
<td>121</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>MACLIB</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>MACRO</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>MEMO</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>M</td>
</tr>
<tr>
<td>NAMES</td>
<td>OFF</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>M</td>
</tr>
<tr>
<td>NOTEBOOK</td>
<td>OFF</td>
<td>132</td>
<td>132</td>
<td>132</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>M</td>
</tr>
<tr>
<td>NETLOG</td>
<td>OFF</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>M</td>
</tr>
<tr>
<td>PLI</td>
<td>ON</td>
<td>72</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>PLIOPT</td>
<td>ON</td>
<td>72</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>OFF</td>
<td>132</td>
<td>132</td>
<td>132</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>M</td>
</tr>
<tr>
<td>UPDATE</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>UPDTXXXX</td>
<td>ON</td>
<td>71</td>
<td>80</td>
<td>72</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>VSBASIC</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>VSBDATA</td>
<td>OFF</td>
<td>132</td>
<td>132</td>
<td>132</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>XEDIT</td>
<td>OFF</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>V</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
<tr>
<td>All others</td>
<td>OFF</td>
<td>80</td>
<td>80</td>
<td>screen width*</td>
<td>F</td>
<td>blank</td>
<td>/</td>
<td>U</td>
</tr>
</tbody>
</table>

* If LRECL is less than the screen width, VERIFY = LRECL. Otherwise, VERIFY = screen width.

Figure A-1. Default Settings According to Filetype (Part 1 of 2)
<table>
<thead>
<tr>
<th>Filetype</th>
<th>Tab Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM3705</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>AMSERV</td>
<td>2 5 10 15 20 25 30 35 40 45 50 55 60</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>BASDATA</td>
<td>7 10 15 20 25 30 80</td>
</tr>
<tr>
<td>BASIC</td>
<td>7 10 15 20 25 30 80</td>
</tr>
<tr>
<td>CNTRL</td>
<td>1 5 8 17 27 31</td>
</tr>
<tr>
<td>COBOL</td>
<td>1 8 12 20 28 36 44 68 72 80</td>
</tr>
<tr>
<td>COPY</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>DIRECT</td>
<td>1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
</tr>
<tr>
<td>EXEC</td>
<td>1 5 8 17 27 31</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>1 7 10 15 20 25 30 80</td>
</tr>
<tr>
<td>JOB</td>
<td>1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
</tr>
<tr>
<td>LISTING</td>
<td>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</td>
</tr>
<tr>
<td>MACLIB</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>MACRO</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>MEMO</td>
<td>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</td>
</tr>
<tr>
<td>NAMES</td>
<td>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</td>
</tr>
<tr>
<td>NETLOG</td>
<td>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</td>
</tr>
<tr>
<td>NOTEBOOK</td>
<td>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</td>
</tr>
<tr>
<td>PLI</td>
<td>2 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80</td>
</tr>
<tr>
<td>PLIOPT</td>
<td>2 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>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</td>
</tr>
<tr>
<td>UPDATE</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>UPDTXXXX</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>VSBASIC</td>
<td>7 10 15 20 25 30 80</td>
</tr>
<tr>
<td>VSBDATA</td>
<td>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</td>
</tr>
<tr>
<td>XEDIT</td>
<td>1 10 16 30 35 40 45 50 55 60 65 70</td>
</tr>
<tr>
<td>All others</td>
<td>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</td>
</tr>
</tbody>
</table>

Figure A-1. Default Settings According to Filetype (Part 2 of 2)
Appendix B:  CMS Editor (EDIT) Compatibility Mode

To edit a file in EDIT compatibility 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 compatibility 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 compatibility 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 compatibility mode) for a particular file, you can specify "OLD" as an option on the EDIT command. For example:

```
EDIT fn ft (OLD)
```

If you want to invoke the old CMS editor each time you issue an EDIT command, see Usage Note 8, below.

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 compatibility mode the same way they are executed under the CMS editor:

```
ALTER   FNAME   PRESERVE   SHORT
AUTO SAVE FORMAT PROMPT STACK
BACKWARD FORWARD QUIT TABSET
BOTTOM   GETFILE RECFM TOP
CASE     IMAGE   RENUM TRUNC
CHANGE   INPUT   REPEAT TYPE
CMS      LINEMODE REPLACE UP
DELETE   LOCATE RESTORE X,Y
DOWN     LONG RETURN ZONE
DSTRING  NEXT REUSE ?
FIND     OVERLAY SAVE $XXXX (macros)
FMODE    SERIAL
```

2. The following EDIT subcommands are executed slightly differently in EDIT compatibility mode:

```
SCROLL/SCROLLUP
Under the CMS editor, these subcommands scroll the file one full screen. In EDIT compatibility 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 compatibility 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 compatibility mode.

4. In EDIT compatibility 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 filemode is ".*' instead of "AI".

6. When you issue an EDIT command, the XEDIT profile macro (if any) is executed automatically.

7. 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); and the column number of the current column (COLUMN=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.

8. When you issue an EDIT command, an EXEC named EDIT EXEC S2 is executed. To cause the EDIT command to invoke the old CMS editor, you must rename this EXEC.

Notes for Macro Writers:
When an EDIT command is issued from an EXEC file, the CMS editor is invoked. To invoke EDIT compatibility mode, you must issue the following statement in your EXEC file:

```
EXEC EDIT ...
```
Figure C-1 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>
<thead>
<tr>
<th>If you used this EDIT command:</th>
<th>Now use this XEDIT command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER</td>
<td>ALTER</td>
</tr>
<tr>
<td>AUTOSAVE</td>
<td>SET AUTOSAVE</td>
</tr>
<tr>
<td>BACKWARD</td>
<td>UP</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>BOTTOM</td>
</tr>
<tr>
<td>CASE</td>
<td>SET CASE</td>
</tr>
<tr>
<td>CHANGE</td>
<td>CHANGE (G not supported)</td>
</tr>
<tr>
<td>CMS</td>
<td>CMS</td>
</tr>
<tr>
<td>DELETE</td>
<td>DELETE</td>
</tr>
<tr>
<td>DOWN</td>
<td>DOWN</td>
</tr>
<tr>
<td>DSTRING</td>
<td>DELETE</td>
</tr>
<tr>
<td>FILE</td>
<td>FILE</td>
</tr>
<tr>
<td>FMODE</td>
<td>SET FMODE</td>
</tr>
<tr>
<td>FNAME</td>
<td>SET FNAME</td>
</tr>
<tr>
<td>FORMAT</td>
<td>SET TERMINAL</td>
</tr>
<tr>
<td>FORWARD</td>
<td>DOWN</td>
</tr>
<tr>
<td>GETFILE</td>
<td>GET</td>
</tr>
<tr>
<td>IMAGE</td>
<td>SET IMAGE</td>
</tr>
<tr>
<td>INPUT</td>
<td>INPUT</td>
</tr>
<tr>
<td>LINEMODE</td>
<td>Not supported</td>
</tr>
<tr>
<td>LOCATE</td>
<td>LOCATE</td>
</tr>
<tr>
<td>LONG</td>
<td>SET MSGMODE ON LONG</td>
</tr>
<tr>
<td>NEXT</td>
<td>NEXT</td>
</tr>
<tr>
<td>OVERLAY</td>
<td>OVERLAY</td>
</tr>
<tr>
<td>PRESERVE</td>
<td>PRESERVE</td>
</tr>
<tr>
<td>PROMPT</td>
<td>Not supported</td>
</tr>
<tr>
<td>QUIT</td>
<td>QUIT</td>
</tr>
<tr>
<td>RECFM</td>
<td>SET RECFM</td>
</tr>
<tr>
<td>RENUM</td>
<td>RENUM</td>
</tr>
<tr>
<td>REPEAT</td>
<td>REPEAT (repeat any previous subcommands)</td>
</tr>
<tr>
<td>REPLACE</td>
<td>REPLACE</td>
</tr>
<tr>
<td>RESTORE</td>
<td>RESTORE</td>
</tr>
</tbody>
</table>

Figure C-1. EDIT Migration Chart (Part 1 of 2)
<table>
<thead>
<tr>
<th>If you used this EDIT command:</th>
<th>Now use this XEDIT command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN</td>
<td>RETURN</td>
</tr>
<tr>
<td>REUSE (=)</td>
<td>=</td>
</tr>
<tr>
<td>SAVE</td>
<td>SAVE</td>
</tr>
<tr>
<td>SCROLL</td>
<td>FORWARD</td>
</tr>
<tr>
<td>SCROLLUP</td>
<td>BACKWARD</td>
</tr>
<tr>
<td>SERIAL</td>
<td>SET SERIAL</td>
</tr>
<tr>
<td>SHORT</td>
<td>SET MSGMODE ON SHORT</td>
</tr>
<tr>
<td>STACK</td>
<td>STACK (STACK string not supported)</td>
</tr>
<tr>
<td>TABSET</td>
<td>SET TABS</td>
</tr>
<tr>
<td>TOP</td>
<td>TOP</td>
</tr>
<tr>
<td>TRUNC</td>
<td>SET TRUNC</td>
</tr>
<tr>
<td>TYPE</td>
<td>TYPE (second operand not supported)</td>
</tr>
<tr>
<td>UP</td>
<td>UP</td>
</tr>
<tr>
<td>VERIFY</td>
<td>SET VERIFY</td>
</tr>
<tr>
<td>X or Y</td>
<td>Can be done via SET SYNONYM and/or REPEAT</td>
</tr>
<tr>
<td>ZONE</td>
<td>SET ZONE</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>nnnn</td>
<td>:nnnn (nnnn is equivalent to +nnnn)</td>
</tr>
<tr>
<td>$DUP</td>
<td>DUPLICATE</td>
</tr>
<tr>
<td>$MOVE</td>
<td>MOVE</td>
</tr>
</tbody>
</table>

Figure C-1. EDIT Migration Chart (Part 2 of 2)
Appendix D: Display Editing System (EDGAR) Compatibility Mode

To edit a file in EDGAR compatibility mode, you must be using a display terminal in full screen mode. You must convert your EDGAR $PROFILE file (one time only). Then, when you issue the EDGAR command, the XEDIT editor automatically places you in EDGAR compatibility mode, in which you can issue both EDGAR and XEDIT commands.

If you want to invoke the original EDGAR (instead of XEDIT in EDGAR compatibility mode) for a particular file, you can specify "OLD" as an option on the EDGAR command. For example:

```plaintext
EDGAR fn ft fm OLD
```

If you want to invoke the original Display Editing System IUP each time you issue an EDGAR command, see Usage Note 6, below.

Converting Your EDGAR $PROFILE File

A default profile is provided. Its filename and filetype are $EDGAR XEDIT. If you have your own EDGAR profile, you must convert your EDGAR $PROFILE file to an XEDIT macro. This conversion needs to be done only once.

The following EXEC converts a file named "filename $PROFILE" to an XEDIT macro named "$filename XEDIT A2".

```plaintext
CNV$PROF [EDGAR] filename
```

where "filename" is the name of your $PROFILE file. If you omit "filename", the default is "EDGAR" (which is usually the name of the $PROFILE file).

Once the $PROFILE file is converted, you can edit a file in EDGAR compatibility mode by issuing the same EDGAR command that you normally use.

Usage Notes:

1. The screen differs from the EDGAR screen in the following ways:
   a. The Type III area contains five equals signs (=====) instead of the EDGAR pattern (*=*=*=). The Type III area of the current line also contains five equals signs; however, the current line is highlighted.
   b. The header line is protected; therefore, you cannot modify the fileid by typing over the header. Instead, you can use the XEDIT subcommands SET FNAME, SET FTYPE, and SET FMODE. The header line does not contain the alteration count (ALT) or the AUTOSAVE filename. (You can use the XEDIT subcommand QUERY AUTOSAVE to display this information.)
   c. A file line that is longer than a screen line is displayed on as many lines as necessary.

2. You may issue both EDGAR and XEDIT commands in EDGAR compatibility mode. If an EDGAR and XEDIT command name are the same, the EDGAR command is executed. To execute the XEDIT command instead, type a single quote (' ) before the command name; for example, 'QUERY AUTOSAVE causes the XEDIT command to be executed.
3. The following EDGAR commands are executed in EDGAR compatibility mode the same way they are executed under EDGAR:

```
CANCEL   QUIT
ENTER    SAVE
FILE
ADD      LOCATE
BACKWARD LOCATE (negative) TABS
BOTTOM   MOVE    TOP
CASE     PFn     UFIND
CLINE    PUTFILE ULOCATE
CMS      QUERY   UP
COPY     RECFM   USEARCH
CP       SEARCH  VIEW
DELETE   /       ZONE
DOWN     SERIAL  .XXXX
DSPC     SET NULLS :XXXX
DSPF     SET AUTOSAVE $XXXX
DUP      SET NUMBERS nnnn
FIND     SET CONVERT $, =, ?
FORWARD  SHIFT  
GETFILE  STACK
```

Type III Commands: A, D, /, "

The following EDGAR $PROFILE functions are supported: FILE, *, PFxx

In addition, the following functions are supported:

```
ECOMMAND
EDGARSVC
```

4. The following subcommands are executed differently in EDGAR compatibility mode:

a. SCREEN

A logical screen must contain at least five lines.

b. FORMAT

When the FORMAT command is used to create multiple views of the same file, each view has its own file identification line (as well as its own command line).

The FORMAT subcommand cannot be used to assign more than one physical screen line for each logical file line. A logical line is displayed on as many physical lines as are required.

c. REPEAT

You cannot use this subcommand in compatibility mode. However, you can perform the same function by issuing a command to change the line and then issuing the XEDIT subcommand REPEAT. The REPEAT subcommand advances the line pointer (in a forward or backward direction) and executes the last subcommand that was entered, for a specified number of times. For example:

```
OVERLAY text
REPEAT 5
```

The OVERLAY subcommand is executed, and then it is repeated on the next five lines.
d. **SET DBLANK**

When **SET DBLANK ON** is in effect, blank lines are removed from a file when a **FILE** or **SAVE** subcommand is issued. They are removed from the copy of the file in virtual storage, and they are not written on the copy of the file on disk.

e. **Naming lines (POINT, .XXXX, :XXXX)**

1. You cannot assign multiple names to a line. When a name is assigned to a line, it replaces the previous one, if any.
2. To name a line from the Type III area, do not use the "<>" symbol at the end of the name; simply enter the name, preceded by a period (.XXXX).
3. You cannot issue the :XXXX< command from the Type III area. Although you can assign both temporary names (.XXXX) and permanent names (:XXXX) to lines, you can assign permanent names only by using the POINT command.

f. **CHANGE (VER option)**

When the **VER** option is specified, the **CHANGE** command is executed. All changed lines are displayed on as many screens as are required.

g. **INPUT, INSERT, REPLACE**

Blank lines entered between lines containing data are kept in the file.

5. The following **EDGAR $PROFILE** functions are handled differently in **EDGAR compatibility mode**:

a. **TABCHR** - The tab character is defined by a **CMS SET INPUT** command (**CMS SET INPUT** char 05). The way tab characters are handled depends on the setting of the **XEDIT** subcommand **SET IMAGE**: if **SET IMAGE ON** is in effect (the default for most filetypes except SCRIPT), a tab character is expanded with blank or filler characters; if **SET IMAGE OFF** or **SET IMAGE CANON** is in effect, tab characters are left as they are entered.

b. **TABDEF** - The limit is one "fill" character.

c. **CMDSYN** - Primitive **EDGAR** functions (for example, **DOWN**) cannot be redefined.

6. When you issue an **EDGAR** command, an **EXEC** named **EDGAR EXEC S2** is executed. To cause the **EDGAR** command to invoke the original **Display Editing System IUP** (the "real" **EDGAR**), instead, you must rename this **EXEC**.

7. If you want to invoke **EDGAR** from within **XEDIT**, you must use the **CMS** subcommand. This command would have the form "CMS EDGAR fn ft".

**Notes for Macro Writers:**

1. If you issue /SOS TABF or /SOS TABB, the cursor is tabbed in both the file lines and in the command line.
2. Because of slight differences in the appearance of the screen, you should not rely on details of the layout in terms of column numbers. In addition, the fileid in the header line cannot be overwritten; you can use the **XEDIT** subcommands **SET FNAME**, **SET FTYPE**, and **SET FMODE** instead.
3. The **SET NOREAD** command is not supported. Instead of redefining the meaning of **SOS ATTN**, you can perform the two functions directly: either stack a null line or issue an **SOS ENTER**.
4. You can edit 26 files simultaneously, that is, when editing multiple files, the number of files in the ring of files in storage is limited to 26.
5. Remember that XEDIT prefix subcommands C, M, F, and P can cause a "pending" status.

6. The XEDIT profile macro, if any, is not executed when you issue the EDGAR command.

Error Messages:

002E FILE {'$EDGAR XEDIT *' | 'EDGAR MODULE *'} NOT FOUND., RC=28

Return Codes:

28 The EDGAR profile or EDGAR module was not found but is required.
Appendix E: Migrating from EDGAR to XEDIT

Figure E-1 lists the EDGAR 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>
<thead>
<tr>
<th>If you used this EDGAR command:</th>
<th>Now use this XEDIT command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCEL</td>
<td>CANCEL</td>
</tr>
<tr>
<td>CLINE</td>
<td>SET CURLINE</td>
</tr>
<tr>
<td>ENTER</td>
<td>XEDIT</td>
</tr>
<tr>
<td>FILE</td>
<td>FILE</td>
</tr>
<tr>
<td>FORMAT</td>
<td>SET SCREEN</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>QUIT</td>
<td>QUIT</td>
</tr>
<tr>
<td>SCREEN</td>
<td>SET SCREEN</td>
</tr>
<tr>
<td>SAVE</td>
<td>SAVE</td>
</tr>
<tr>
<td>ADD</td>
<td>ADD</td>
</tr>
<tr>
<td>BACK</td>
<td>BACKWARD</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>BOTTOM</td>
</tr>
<tr>
<td>CASE</td>
<td>SET CASE</td>
</tr>
<tr>
<td>CHANGE</td>
<td>CHANGE</td>
</tr>
<tr>
<td>CMS</td>
<td>CMS</td>
</tr>
<tr>
<td>COPY</td>
<td>COPY</td>
</tr>
<tr>
<td>CP</td>
<td>CP</td>
</tr>
<tr>
<td>DELETE</td>
<td>DELETE</td>
</tr>
<tr>
<td>DOWN</td>
<td>UP</td>
</tr>
<tr>
<td>DSPC</td>
<td>QUERY CURSOR</td>
</tr>
<tr>
<td>DSPF</td>
<td>QUERY PfN</td>
</tr>
<tr>
<td>DUP</td>
<td>Duplicat</td>
</tr>
<tr>
<td>FIND</td>
<td>FIND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FORWARD</td>
<td>FORWARD</td>
</tr>
<tr>
<td>GETFILE</td>
<td>GET</td>
</tr>
<tr>
<td>INPUT</td>
<td>INPUT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>INSERT</td>
<td>Not supported</td>
</tr>
<tr>
<td>LEFT</td>
<td>RIGHT</td>
</tr>
<tr>
<td>LOCATE</td>
<td>LOCATE</td>
</tr>
<tr>
<td>MOVE</td>
<td>MOVE</td>
</tr>
</tbody>
</table>

Figure E-1. EDGAR Migration Chart (Part 1 of 2)
<table>
<thead>
<tr>
<th>If you used this EDGAR command:</th>
<th>Now use this XEDIT command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFn</td>
<td>SET PFn</td>
</tr>
<tr>
<td>POINT</td>
<td>SET POINT</td>
</tr>
<tr>
<td>PUTFILE</td>
<td>PUT</td>
</tr>
<tr>
<td>QUERY</td>
<td>QUERY</td>
</tr>
<tr>
<td>RECFM</td>
<td>SET RECFM</td>
</tr>
<tr>
<td>REPEAT</td>
<td>Can be done via OVERLAY (followed by REPEAT)</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Can be done via DELETE and INPUT</td>
</tr>
<tr>
<td>RIGHT</td>
<td>LEFT</td>
</tr>
<tr>
<td>SEARCH</td>
<td>LOCATE with SET WRAP ON</td>
</tr>
<tr>
<td>SERIAL</td>
<td>SET SERIAL</td>
</tr>
<tr>
<td>SET</td>
<td>SET (DBLANK, NOREAD not supported)</td>
</tr>
<tr>
<td>SHIFT</td>
<td>SHIFT</td>
</tr>
<tr>
<td>STACK</td>
<td>STACK</td>
</tr>
<tr>
<td>TABS</td>
<td>SET TABS</td>
</tr>
<tr>
<td>TOP</td>
<td>TOP</td>
</tr>
<tr>
<td>/string</td>
<td>LOCATE -/string/</td>
</tr>
<tr>
<td>UP</td>
<td>DOWN</td>
</tr>
<tr>
<td>USEARCH</td>
<td>LOCATE -/string/ with SET WRAP ON</td>
</tr>
<tr>
<td>VIEW</td>
<td>SET VERIFY</td>
</tr>
<tr>
<td>ZONE</td>
<td>SET ZONE</td>
</tr>
<tr>
<td>.XXX</td>
<td>.XXX (target facility)</td>
</tr>
<tr>
<td>$XXXX</td>
<td>$XXXX (execute $XXXX EXEC instead of XXXX EXEC)</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>.XXX</td>
<td>.XXX</td>
</tr>
</tbody>
</table>

Type III:

Figure E-1. EDGAR Migration Chart (Part 2 of 2)
Appendix F: 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
- EDGAR
- JOIN
- SCHANGE
- SPLIT
- 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).

```plaintext
*OPTIMIZED AT 14:13:12 ON 80/01/29
** NOTICE:
** 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

The format of the VMFOPT macro is as follows:

```
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
&STACK LIPO 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.

The format of the VMFDEOPT macro is as follows:

| 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 G: A Summary of XEDIT Subcommands and Macros

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Add n lines after current line.</td>
</tr>
<tr>
<td>ALTER</td>
<td>Change a single character to another (character or hex).</td>
</tr>
<tr>
<td>BACKWARD</td>
<td>Scroll backward n frames.</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Go to last line of file.</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Terminate all files.</td>
</tr>
<tr>
<td>CAPPEND</td>
<td>Add text to end of current line.</td>
</tr>
<tr>
<td>CDELETE</td>
<td>Delete characters, starting at column pointer.</td>
</tr>
<tr>
<td>CFIRST</td>
<td>Move column pointer to beginning of line (zone).</td>
</tr>
<tr>
<td>CHANGE</td>
<td>Change one string to another.</td>
</tr>
<tr>
<td>CINSERT</td>
<td>Insert text in the current line.</td>
</tr>
<tr>
<td>CLAST</td>
<td>Move the column pointer to the end of the line (zone).</td>
</tr>
<tr>
<td>CLOCATE</td>
<td>Locate a string; move the column pointer and the line pointer.</td>
</tr>
<tr>
<td>CMS</td>
<td>Pass a command to CMS, or enter CMS subset mode.</td>
</tr>
<tr>
<td>CMSG</td>
<td>Display message in command line of user's screen.</td>
</tr>
<tr>
<td>COMMAND</td>
<td>Execute a subcommand without checking for synonym or macro.</td>
</tr>
<tr>
<td>COMPRESS</td>
<td>Prepare line(s) for realignment by replacing blanks with tab characters.</td>
</tr>
<tr>
<td>COPY</td>
<td>Copy line(s) at specified location.</td>
</tr>
<tr>
<td>COUNT</td>
<td>Display the number of times a string appears.</td>
</tr>
<tr>
<td>COVERLAY</td>
<td>Replace characters, starting at column pointer.</td>
</tr>
<tr>
<td>CP</td>
<td>Pass command to VM/SP control program.</td>
</tr>
<tr>
<td>CREPLACE</td>
<td>Replace characters, starting at the column pointer.</td>
</tr>
<tr>
<td>CURSOR</td>
<td>Move the cursor to specified position on the screen.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete line(s).</td>
</tr>
<tr>
<td>DOWN</td>
<td>Move line pointer n lines toward end of file (same as NEXT).</td>
</tr>
<tr>
<td>DUPLICATE</td>
<td>Duplicate line(s).</td>
</tr>
<tr>
<td>EMSG</td>
<td>Display a message and sound the alarm.</td>
</tr>
<tr>
<td>EXPAND</td>
<td>Reposition data according to new tab settings.</td>
</tr>
<tr>
<td>FILE</td>
<td>Write file on disk.</td>
</tr>
<tr>
<td>FIND</td>
<td>Search for line that starts with specified text.</td>
</tr>
<tr>
<td>FINDUP</td>
<td>Search for a line that starts with specified text; searches in a backward direction.</td>
</tr>
<tr>
<td>FORWARD</td>
<td>Scroll forward n frames.</td>
</tr>
<tr>
<td>GET</td>
<td>Insert lines from another file.</td>
</tr>
<tr>
<td>HELP</td>
<td>Request on-line display of XEDIT subcommands and macros; invoke the CMS HELP facility.</td>
</tr>
<tr>
<td>HEXTYPE</td>
<td>Display line(s) in hexadecimal and EBCDIC.</td>
</tr>
<tr>
<td>INPUT</td>
<td>Insert a single line, or enter input mode.</td>
</tr>
<tr>
<td>Subcommand</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Join</td>
<td>Join lines.</td>
</tr>
<tr>
<td>LEFT</td>
<td>View data to the left of column one.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Read file into storage; use in profile macro only.</td>
</tr>
<tr>
<td>Locate</td>
<td>Move line pointer to specified target.</td>
</tr>
<tr>
<td>LOWercas</td>
<td>Change uppercase letters to lowercase.</td>
</tr>
<tr>
<td>MACRO</td>
<td>Execute macro without checking for subcommand or synonym.</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Display a SET subcommand current values in the command line, so it can be overtyped and reentered.</td>
</tr>
<tr>
<td>MOVE</td>
<td>Move line(s) to another place in the file.</td>
</tr>
<tr>
<td>MSG</td>
<td>Display message in message line.</td>
</tr>
<tr>
<td>Next</td>
<td>Move line pointer n lines toward end of file (same as DOWN).</td>
</tr>
<tr>
<td>NFind</td>
<td>Search for first line that does not match specified text.</td>
</tr>
<tr>
<td>NFINDUp</td>
<td>Search backward for first line that does not match specified text.</td>
</tr>
<tr>
<td>Overlay</td>
<td>Replace characters in current line.</td>
</tr>
<tr>
<td>PARSE</td>
<td>Scans a line of a macro to check the format of its operands.</td>
</tr>
<tr>
<td>POWERinp</td>
<td>Enter an input mode for continuous typing.</td>
</tr>
<tr>
<td>PREServe</td>
<td>Save settings of variables until RESTORE.</td>
</tr>
<tr>
<td>PURge</td>
<td>Remove macro from virtual storage.</td>
</tr>
<tr>
<td>PUT</td>
<td>Insert lines into another file (new or existing), or into a buffer (to be retrieved by GET from another file).</td>
</tr>
<tr>
<td>PUTD</td>
<td>Insert lines into another file (new or existing) or into a buffer (to be retrieved by GET from another file); delete original lines.</td>
</tr>
<tr>
<td>Query</td>
<td>Display the current value of editing options.</td>
</tr>
<tr>
<td>QUIT</td>
<td>End an editing session without saving changes.</td>
</tr>
<tr>
<td>READ</td>
<td>Place information from the terminal in the console stack.</td>
</tr>
<tr>
<td>RECover</td>
<td>Replace deleted lines.</td>
</tr>
<tr>
<td>RENum</td>
<td>Renumber VS BASIC or FREEFORT file.</td>
</tr>
<tr>
<td>REPeat</td>
<td>Advance line pointer and re-execute last subcommand.</td>
</tr>
<tr>
<td>Replace</td>
<td>Replace current line, or delete current line and enter input mode.</td>
</tr>
<tr>
<td>RESet</td>
<td>Remove prefix subcommands when screen is in &quot;pending&quot; or &quot;incomplete&quot; status.</td>
</tr>
<tr>
<td>RESTore</td>
<td>Restore settings of XEDIT variables to values they had when PRE­SERVE was issued.</td>
</tr>
<tr>
<td>Right</td>
<td>View data to the right of the last (right-most) column.</td>
</tr>
<tr>
<td>SAVE</td>
<td>Write file on disk and remain in edit mode.</td>
</tr>
<tr>
<td>SCHange</td>
<td>Make a selective change, using PF keys.</td>
</tr>
<tr>
<td>SET APL</td>
<td>Inform the editor if APL keys are used.</td>
</tr>
<tr>
<td>SET ARBchar</td>
<td>Define an arbitrary character, which allows you to specify only the beginning and the end of a character string that is the object of a target search.</td>
</tr>
<tr>
<td>SET Autosave</td>
<td>Automatically issue a SAVE subcommand at specified intervals.</td>
</tr>
<tr>
<td>SET CASE</td>
<td>Upper or lower case control; specify if case is significant in target searches.</td>
</tr>
<tr>
<td>Subcommand</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SET CMDline</td>
<td>Move the position of the command line.</td>
</tr>
<tr>
<td>SET COLPtr</td>
<td>Specify if column pointer is displayed (typewriter terminals only).</td>
</tr>
<tr>
<td>SET CTLchar</td>
<td>Define control character(s) to specify that fields be displayed highlighted or not, protected or not, visible or not.</td>
</tr>
<tr>
<td>SET CURLine</td>
<td>Define the position of the current line on the screen.</td>
</tr>
<tr>
<td>SET ESCape</td>
<td>Define a character that allows you to enter a subcommand while in input mode (typewriter terminals only).</td>
</tr>
<tr>
<td>SET FILLer</td>
<td>Define a character that is used when a line is expanded.</td>
</tr>
<tr>
<td>SET FMode</td>
<td>Change the filemode of the current file.</td>
</tr>
<tr>
<td>SET FName</td>
<td>Change the filename of the current file.</td>
</tr>
<tr>
<td>SET FType</td>
<td>Change the filetype of the current file.</td>
</tr>
<tr>
<td>SET HEX</td>
<td>Allows string targets to be specified in hexadecimal.</td>
</tr>
<tr>
<td>SET IMAGE</td>
<td>Control how tabs and backspaces are handled.</td>
</tr>
<tr>
<td>SET IMPcmscp</td>
<td>Control whether subcommands not recognized by the editor are transmitted to CMS and CP.</td>
</tr>
<tr>
<td>SET LINENd</td>
<td>Define a line end character.</td>
</tr>
<tr>
<td>SET LRecl</td>
<td>Define a new logical record length.</td>
</tr>
<tr>
<td>SET MACRO</td>
<td>Control the order in which the editor searches for subcommands and macros.</td>
</tr>
<tr>
<td>SET MASK</td>
<td>Define a new mask, which is the contents of added lines and the input zone.</td>
</tr>
<tr>
<td>SET MSGMode</td>
<td>Control the message display.</td>
</tr>
<tr>
<td>SET NONDisp</td>
<td>Define a character that is used in place of non-displayable characters.</td>
</tr>
<tr>
<td>SET NULLs</td>
<td>Specify whether trailing blanks are replaced with nulls to allow character insertion.</td>
</tr>
<tr>
<td>SET NUMBER</td>
<td>Specify whether file line numbers are displayed in the prefix area.</td>
</tr>
<tr>
<td>SET PACK</td>
<td>Specify if the file is to be written to disk in packed format.</td>
</tr>
<tr>
<td>SET PFn</td>
<td>Define a meaning for a PF key.</td>
</tr>
<tr>
<td>SET Point</td>
<td>Define a symbolic name for the current line.</td>
</tr>
<tr>
<td>SET PREFIX</td>
<td>Control the display of the prefix area; define a synonym for a prefix subcommand.</td>
</tr>
<tr>
<td>SET RANGE</td>
<td>Define a new &quot;top&quot; and &quot;bottom&quot; for the file.</td>
</tr>
<tr>
<td>SET RECFm</td>
<td>Define the record format.</td>
</tr>
<tr>
<td>SET RESERVed</td>
<td>Reserve a line, which cannot be used by the editor.</td>
</tr>
<tr>
<td>SET SCALe</td>
<td>Control the display of the scale line.</td>
</tr>
<tr>
<td>SET SCReen</td>
<td>Divide the screen into logical screens, for multiple views of the same or of different files.</td>
</tr>
<tr>
<td>SET SERIAL</td>
<td>Control file serialization.</td>
</tr>
<tr>
<td>SET SPAN</td>
<td>Allows a string target to span a number of lines.</td>
</tr>
<tr>
<td>SET STAY</td>
<td>Specify whether the line pointer moves when a string is not located.</td>
</tr>
<tr>
<td>SET STREAmp</td>
<td>Specify whether the editor searches only the current line or the whole file for a column-target.</td>
</tr>
<tr>
<td>SET SYNonym</td>
<td>Specify whether the editor looks for synonyms; assign a synonym.</td>
</tr>
<tr>
<td>SET TABLine</td>
<td>Control the display of the tab line.</td>
</tr>
<tr>
<td>SET TABS</td>
<td>Define the logical tab stops.</td>
</tr>
<tr>
<td>Subcommand</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SET TERMINal</td>
<td>Specify whether a terminal is used in line mode or full screen mode.</td>
</tr>
<tr>
<td>SET TEXT</td>
<td>Inform the editor if TEXT keys are used.</td>
</tr>
<tr>
<td>SET TOEOF</td>
<td>Control the display of TOF/EOF lines.</td>
</tr>
<tr>
<td>SET TRunc</td>
<td>Define the truncation column.</td>
</tr>
<tr>
<td>SET VARblank</td>
<td>Specify whether the number of blanks between two words is significant in a target search.</td>
</tr>
<tr>
<td>SET Verify</td>
<td>Control whether lines changed by subcommands are displayed; define the columns displayed and whether displayed in EBCDIC or hexadecimal or both.</td>
</tr>
<tr>
<td>SET WRap</td>
<td>Control whether the editor wraps around the file if EOF is reached during a search.</td>
</tr>
<tr>
<td>SET Zone</td>
<td>Define new limits within each line for target searches.</td>
</tr>
<tr>
<td>SET =</td>
<td>Insert string into the equal buffer.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Move data right or left (data loss possible).</td>
</tr>
<tr>
<td>SORT</td>
<td>Sort all or part of a file, in ascending or descending order.</td>
</tr>
<tr>
<td>SOS</td>
<td>Specify functions for screen operation simulation.</td>
</tr>
<tr>
<td>SPLit</td>
<td>Split a line into two or more lines.</td>
</tr>
<tr>
<td>STAck</td>
<td>Place line(s) from the file into the console stack.</td>
</tr>
<tr>
<td>STATus</td>
<td>Display SET subcommand current settings; create a macro that contains these settings.</td>
</tr>
<tr>
<td>TOP</td>
<td>Move line pointer to null TOP OF FILE line.</td>
</tr>
<tr>
<td>TRAnsf</td>
<td>Place editing variable(s) in the console stack, for use by a macro.</td>
</tr>
<tr>
<td>Type</td>
<td>Display lines.</td>
</tr>
<tr>
<td>UP</td>
<td>Move line pointer toward top of file.</td>
</tr>
<tr>
<td>UPPercas</td>
<td>Translate all lowercase characters to uppercase.</td>
</tr>
<tr>
<td>Xedit</td>
<td>Edit multiple files.</td>
</tr>
<tr>
<td>&amp;</td>
<td>Use before a subcommand for repeated execution.</td>
</tr>
<tr>
<td>=</td>
<td>Re-execute the last subcommand or macro.</td>
</tr>
<tr>
<td>?</td>
<td>Display the last subcommand executed.</td>
</tr>
</tbody>
</table>

**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).
- **M**: Move line(s).
- **P**: Move or copy preceding this line.
- **=**: Duplicate line(s).
- **/**: Make this line the current line.
- **SCALE**: Display the scale on this line.
- **TABL**: Display the tab line on this line.
- **.xxxxx**: Assign symbolic name to this line.
Index

backspace character
  affected by SET IMAGE 3-126
BACKWARD subcommand 3-6
  assigned to PF key 3-6
blank
  characters removed by COMPRESS 3-25
  separating file lines during string target search 3-153
blanks between words, determining significance of 3-166
block of lines
  copying 4-4
  deleting 4-5
  duplicating 4-16
  moving 4-10
bottom of range 3-144
BOTTOM subcommand 3-7

C prefix subcommand 4-4
CANCEL macro 3-8
canonical order
  specified by SET IMAGE 3-126
CAPPEND macro 3-9
  example of 3-9
case
  ignoring difference in target search 3-116
  respecting difference in target search 3-116
  setting 3-116
  translating to lowercase 3-66
  translating to uppercase 3-191
CASE
  option in QUERY 3-88
  option in SET 3-116
  option in TRANSFER 3-183
CDELETE subcommand 3-10
  example of 3-11
CFIRST subcommand 3-12
  affected by zone 3-170
  example of 3-12
change
  global 3-13
  selective 3-108
CHANGE subcommand 3-13
  example of 3-15
  used in selective change 3-108
changed lines
  displayed 3-167
changing characters, using CHANGE 3-13
character
  nondisplayable, defining character
  used in place of 3-135
  specifying in hexadecimal 3-125
character delete, using CDELETE 3-10
character insert, using CINSERT 3-16
character overlay, using COVERLAY 3-32
character replacement, using CREPLACE 3-34
character set usage I-I
CINSERT subcommand 3-16
  example of 3-16
CLAST subcommand 3-17
  example of 3-17
CLOCATE subcommand 3-18
  example of 3-20
  used in selective change 3-108

Index  X - 1
CMDLINE
  option in QUERY 3-89
  option in SET 3-117
  option in TRANSFER 3-184
CMS, transmitting command to 3-21
CMS editor
  compatibility with B-1
  invoking B-1
CMS subcommand 3-21
CMS subset mode
  commands valid in 3-21
  entering 3-21
  returning from 3-21
CMSG subcommand 3-23
code conversion
  of APL keys 3-111
  of TEXT keys 3-163
COLPTR
  option in QUERY 3-89
  option in SET 3-118
  option in TRANSFER 3-184
column number
  specifying column-target as 3-18
column pointer
  displaying on typewriter terminal 3-118
  moved by CFIRST 3-12
  moved by CLAST 3-17
  moved by CLOCATE 3-18
  movement restricted by zone 3-170
  moving 4-15
  removing from typewriter terminal 3-118
  splitting a line at 3-178
  use in CAPPEND 3-9
  use in CDELETE 3-10
  use in CINSERT 3-16
  use in COVERLAY 3-32
  use in CREPLACE 3-34
  use in JOIN 3-55
  use in SPLIT 3-178
column-target
  as absolute column number 3-18
  as complex string expression 3-19
  as relative displacement 3-18
  as string expression 3-18
  description of 3-18
  search for affected by SET STREAM 3-156
  use in CDELETE 3-10
  use in CLOCATE 3-18
COLUMN
  option in QUERY 3-89
  option in TRANSFER 3-184
columns displayed
  multiple pairs of 3-167
  specifying 3-167
command line
  changing position of 3-117
  displaying message in 3-23
  redisplaying subcommand in 3-196
  stacked by READ 3-96
COMMAND subcommand 3-24
commands, transmitted to CMS/CP 3-128
compatibility
  with CMS editor B-1
  with Display Editing System D-1
complex string expression
  specifying column-target as 3-19
  specifying target as 3-63

COMPRESS subcommand 3-25
  example of 3-26
  compressing records
    using SET PACK 3-138
  concatenation, of lines during string target search 3-153
  console stack
    used by PARSE 3-78, 3-79
    used by READ 3-96
    used by STACK 3-180
    used by TRANSFER 3-183
  continuous typing 3-80
  control character 3-118.1
  conventions, notation 1-2
  converting EDGAR SPROFILE D-1
COPY subcommand 3-28
  example of 3-29
  copying block of lines 4-4
  copying lines
    using C (prefix subcommand) 4-4
    using COPY 3-28
COUNT subcommand 3-30
  example of 3-31
  counting a string 3-30
COVERLAY subcommand 3-32
  example of 3-32
  CP, transmitting command to 3-33
  CP console function mode 3-33
  CP subcommand 3-33
  creating a file
    using PUT 3-84
    using PUTD 3-86
    using XEDIT command 2-1
    using XEDIT subcommand 3-192
CREPLACE subcommand 3-34
  compared to OVERLAY 3-77
CTLCHAR
  option in QUERY 3-89
  option in SET 3-118.1
  option in TRANSFER 3-184
CURLINE
  option in QUERY 3-89
  option in SET 3-119
  option in TRANSFER 3-184
  current line
    advancing
      using a target 3-62
      using DOWN 3-39
      using NEXT 3-73
    appending text to 3-9
    changing position on screen of 3-119
    defining line on screen as 3-119
    replacing 3-102
    setting 4-15
cursor
  displaying position of 3-36, 3-89
  joining lines at 3-55
  moving in the file 3-35
  moving on the screen 3-35
  moving to command line 3-35
  moving to current column 3-35
  splitting a line at 3-178
  transferring position of 3-36, 3-184
CURSOR
  option in QUERY 3-89
  option in TRANSFER 3-184
CURSOR subcommand 3-35
D prefix subcommand 4-5
example of 4-6
defaults, according to filetype A-I
DELETE subcommand 3-37
example of 3-38
deleted lines, recovering 3-98
deleting characters 3-10
deleting lines
using D (prefix subcommand) 4-5
using DELETE 3-37
deoptimizing macro F-2
DES (see Display Editing System)
descending order, sorting in 3-174
destination
of copied lines
specifying 4-4
specifying using F 4-8
specifying using P 4-12
of moved lines
specifying 4-10
specifying using F 4-8
specifying using P 4-12
disconnect, full screen switched to line mode after 3-162
Display Editing System
compatibility with D-1
invoking D-1, D-3
subcommands supported D-2
display-terminal
using in full screen mode 3-162
using in line mode 3-162
displaying
changed lines 3-167
data in hexadecimal 3-167
line numbers on screen 3-137
lines using TYPE 3-188
message
using CMSG 3-23
using EMSG 3-41
using MSG 3-72
setting of editing options 3-88
DOWN subcommand 3-39
DUPLICAT subcommand 3-40
duplicating block of lines 4
duplicating lines
using " (prefix subcommand) 4-16
using DUPLICAT 3-40
E prefix subcommand 4-7
EBCDIC order, sorting in 3-174
EDGAR (see Display Editing System)
EDGAR $PROFILE, converting D-1
EDIT compatibility mode B-1
EDIT subcommands, supported in compatibility mode B-1
editing multiple files 2-3, 3-149
editing options
displaying setting of 3-88
transferring setting of 3-183
editing variables
restoring 3-104
saving 3-82
editor, invoking 2-1
EMSG subcommand 3-41
END OF FILE line
controlling display of 3-164
END OF RANGE line
controlling display of 3-164

ending editing session
using CANCEL 3-8
using FILE 3-44
using QUIT 3-94
entering subcommands, rules for 1-1
EOF
option in QUERY 3-89
option in TRANSFER 3-184
equal buffer, inserting string in 3-172
error message
displaying 3-41
HELP display of 3-51
escape character
used on typewriter terminal 3-120
ESCAPE
option in QUERY 3-89
option in SET 3-120
option in TRANSFER 3-184
EXPAND subcommand 3-42
example of 3-26
used with COMPRESS 3-26
expanding data 3-42
expanding tabs
filler character used in 3-121
extending a line 4-7

F prefix subcommand 4-8
example of 4-11
file identifier
changing
using FILE 3-43
using SAVE 3-107
of AUTOSAVE file 3-114
file lines, stacking 3-180
file serialization 3-151
FILE subcommand 3-43
filemode
changing 3-122
filename
changing 3-123
files
transferring data between
using GET 3-49
using PUT 3-84
using PUTD 3-86
filetype
changing 3-124
defaults according to A-I
filler character
defining 3-121
removed by COMPRESS 3-25
FILLER
option in QUERY 3-89
option in SET 3-121
option in TRANSFER 3-184
FIND subcommand 3-45
example of 3-43
finding data 3-45, 3-47
FINDUP subcommand 3-47
fixed packed record format 3-145
fixed record format 3-145
FMODE
option in QUERY 3-89
option in SET 3-122
option in TRANSFER 3-184

Index  X - 3
FNAME
option in QUERY 3-89
option in SET 3-123
option in TRANSFER 3-184
following, line as destination 4-8
FORWARD subcommand 3-48
assigned to PF key 3-48
FREEFORT file-
renumbering 3-100
FTYPE
option in QUERY 3-89
option in SET 3-124
option in TRANSFER 3-184
full screen mode. editing in 3-162
GET subcommand 3-49
example of 3-50
getting a file 3-49
global change 3-13
help display 3-51
help menus 3-51
HELP subcommand 3-51
HEX
option in QUERY 3-89
option in SET 3-125
option in TRANSFER 3-184
hexadecimal
displaying in
using HEXTYPE 3-52
using SET VERIFY 3-167
recognizing operands specified in 3-125
HEXTYPE macro 3-52
example of 3-52
I prefix subcommand 4-9
ignoring case difference 3-116
IMAGE
option in QUERY 3-89
option in SET 3-126
option in TRANSFER 3-184
IMPCMSCP
option in QUERY 3-89
option in SET 3-128
option in TRANSFER 3-184
implied transmission to CMS/CP 3-128
initial settings
of PF keys 3-139
of SET options 3-110
(see also each SET option description)
input mode
entered using INPUT 3-53
entered using REPLACE 3-102
entering subcommand in 3-120
screen layout in 3-53
using PF keys in 3-54
INPUT subcommand 3-53
input zone
area of screen 3-53
changing size of 3-53
insert key
using in power typing 3-80
using with SET NULLS ON 3-136
inserting
a single line
using INPUT 3-53
inserting a file
using GET 3-49
using PUT 3-84
using PUTD 3-86
inserting characters
using CINSERT 3-16
using PA2 key 3-136
using SET NULLS ON 3-136
inserting lines
using I (prefix subcommand) 4-9
inserting part of a file
using GET 3-49
using PUT 3-84
using PUTD 3-86
JOIN macro 3-55
example of 3-56
joining lines
at column number 3-55
at column pointer 3-55
at cursor 3-55
with strings inserted 3-55
last subcommand
advancing line pointer and repeating 3-101
displaying 3-196
re-executing 3-195
LASTMSG
option in QUERY 3-89
option in TRANSFER 3-184
left shift 3-173
LEFT subcommand 3-57
example of 3-58
LENGTH
option in QUERY 3-89
option in TRANSFER 3-184
limits
for column pointer movement, defining 3-170
for line pointer movement, defining 3-144
LINE
option in QUERY 3-89
option in TRANSFER 3-184
line end character
defining 3-129
recognizing 3-129
used in power typing 3-80
line mode, editing in 3-162
line name
assigning using .xxxx (prefix subcommand) 4-17
assigning using SET POINT 3-141
deleting 3-141
specifying target as 3-62
line number
specifying target as 3-62
line numbers
displaying 3-137
renumbering 3-100
line pointer
advancing
using DOWN 3-39
using NEXT 3-73
using target 3-62
advancing and repeating last subcommand 3-101
controlling movement of when string not found 3-155
effect of SET STAY on 3-155
moving to last file line 3-7
moving to TOF 3-182
moving up 3-189

X - 4 IBM VM/SP: System Product Editor Command and Macro Reference
moving
using / (prefix subcommand) 4-15
line pointer movement
defining new limits for 3-144
LINEND
option in QUERY 3-89
option in SET 3-129
option in TRANSFER 3-184
LOAD subcommand 3-59
LOCATE subcommand 3-62
locating
using CLOCATE 3-18
using LOCATE 3-62
logical line, extending 4-7
logical record length, defining 3-130
logical screen 3-149
logical tab stops, defining 3-161
LOWERCASE subcommand 3-66
example of 3-66
lowercase, translating characters to 3-66
LRECL
option in QUERY 3-89
option in SET 3-130
option in TRANSFER 3-89
LSCREEN
option in QUERY 3-90
option in TRANSFER 3-185

M prefix subcommand 4-10
example of 4-11
macro
containing SET options
creating 3-181
controlling search order for 3-131
deoptimizing F-2
executing alphanumeric macro name 3-67
executing without subcommand or synonym check 3-67
optimizing F-1
removing copy from storage 3-83
reserving a line for use by 3-146
scanning format of 3-78
MACRO
option in QUERY 3-90
option in SET 3-131
option in TRANSFER 3-185
macro check, overriding with COMMAND 3-24
MACRO subcommand 3-67
macros, list of optimized F-1
MASK
option in QUERY 3-90
option in SET 3-132
option in TRANSFER 3-185
mask line
changing 3-132
defining 3-132
menus, HELP 3-51
message
display controlled by SET MSGMODE 3-134
displayed in command line 3-23
displayed in message line
using EMSG 3-41
using MSG 3-72
severity of 3-41
warning
issued by QUIT 3-94
message identification 3-41

migration
from EDGAR to XEDIT E-1
from EDIT to XEDIT C-1
mixed case, specifying 3-116
MODIFY macro 3-68
example of 3-68
modifying SET values 3-68
MOVE subcommand 3-70
moving block of lines 4-10
moving lines
using M (prefix subcommand) 4-10
using MOVE 3-70
MSG subcommand 3-72
MSGMODE
option in QUERY 3-90
option in SET 3-134
option in TRANSFER 3-185
multiple files
displaying 3-149
depending 3-192
 quitting 3-8
multiple logical screens, defining 3-149
multiple subcommands, entered on command line 3-129

naming a line
using .xxxx (prefix subcommand) 4-17
using SET POINT 3-141
NBFILE
option in QUERY 3-90
option in TRANSFER 3-185
NEXT subcommand 3-73
example of 3-74
NFIND subcommand 3-75
NFINDUP subcommand 3-76
NFU (see NFINDUP)
NONDISP
option in QUERY 3-90
option in SET 3-135
option in TRANSFER 3-185
nondisplayable character, defining character used in place of 3-135
not finding text
using NFIND 3-75
using NFINDUP 3-76
notation conventions 1-2
NULLS
option in QUERY 3-90
option in SET 3-136
option in TRANSFER 3-185
nulls, replacing trailing blanks with 3-136
NUMBER
option in QUERY 3-90
option in SET 3-136
option in TRANSFER 3-185

optimizing macro F-1
OVERLAY subcommand 3-77
overlaying characters
using COVERLAY 3-32
using OVERLAY 3-77

P prefix subcommand 4-12
PACK
option in QUERY 3-90
option in SET 3-138
option in TRANSFER 3-185

Index X - 5
packed file
  defining record format for  3-145
  inserting  3-49
  specifying  3-138
PARSE macro  3-78
PA2 key
  using instead of SET NULLS ON  3-136
PF key
  assigning a sequence of subcommands to  3-139
  defining meaning for  3-139
  removing meaning from  3-139
  value placed in console stack  3-96
PF keys
  used in SCHANGE  3-108
  using in input mode  3-54
PFn
  option in QUERY  3-90
  option in SET  3-139
  option in TRANSFER  3-185
POINT
  option in QUERY  3-90
  option in SET  3-141
  option in TRANSFER  3-185
power typing
  causing a break in data typed in  3-80
  entering data with  3-80
  using a line end character in  3-80
  using the insert key in  3-80
POWERINP subcommand  3-80
  pre-filling line with mask  3-132
preceding, line as destination  4-12
PREFIX
  option in QUERY  3-90
  option in SET  3-143
  option in TRANSFER  3-185
prefix area
  controlling display of  3-143
  entering subcommands in  4-2
  resetting  4-2
prefix subcommand
  .xxxx  4-17
  /  4-15
  "  4-16
  A  4-2
  C  4-4
  D  4-5
  defining synonym for  3-143
  E  4-7
  F  4-8
  I  4-9
  M  4-10
  P  4-12
  removing from screen  3-103
  SCALE  4-13
  TABL  4-14
prefix subcommands
  list of  4-1
  menu display of  3-51
  rules for entering  4-1
PRECREATE subcommand  3-82
printer spool
  copying contents of
  using PF key  3-139
profile macro
  not executing  2-2
  specifying macro name  2-2
  used to prompt for options in  3-59, 3-60
  using LOAD in  3-59
program function key (see PF key)
protected QUIT  3-8, 3-94
PURGE subcommand  3-83
PUT subcommand  3-84
  example of  3-50, 3-85
PUTD subcommand  3-86
QQUIT  3-94
QUERY subcommand  3-88
QUIT subcommand  3-94
  protected  3-8, 3-94
  unprotected  3-8, 3-94
quitting multiple files  3-8
range, defining  3-144
RANGE
  option in QUERY  3-90
  option in SET  3-144
  option in TRANSFER  3-185
re-executing subcommand
  using =  3-195
  using REPEAT  3-101
READ subcommand  3-96
realign data
  example of  3-26
  sequence used to  3-25
RECFM
  option in QUERY  3-90
  option in SET  3-145
  option in TRANSFER  3-185
reconnecting, after disconnect  3-162
record format
  defining  3-145
  fixed
  logical record length of  3-130
  variable
  logical record length of  3-130
RECOVER subcommand  3-98
recoverying deleted lines  3-98
recursive editing  2-3
redisplaying subcommand
  using &  3-194
  using ?  3-196
relative column number
  specifying column-target as  3-18
relative displacement
  specifying target as  3-62
removing prefix subcommands  3-103
RENUM subcommand  3-100
renumbering line numbers
  of VSBASIC or FREEFORT files  3-100
REPEAT subcommand  3-101
repeating last subcommand  3-101
REPLACE subcommand  3-102
replacing characters
  using COVERLAY  3-32
  using CRPELACE  3-34
  current line  3-102
RESERVED
  option in QUERY  3-90
  option in SET  3-146
  option in TRANSFER  3-185
reserved line
  displaying data on 3-146
  displaying the number of 3-146
  returning to the editor 3-146
  transferring the number of 3-146
reserving a line 3-146
RESET subcommand 3-103
respecting case difference 3-116
RESTORE subcommand 3-104
restoring the screen
  after LEFT 3-57
  after RIGHT 3-105
restoring variables 3-104
retrieving
  lines saved by PUT 3-84
  lines saved by PUTD 3-86
  deleted lines 3-98
returning
  from CMS subset mode 3-21
  from CP 3-33
right shift 3-173
RIGHT subcommand 3-105
example of 3-106
RING
  option in QUERY 3-90
  ring of files 3-149, 3-192
rules
  for entering subcommands 1-1
SAVE subcommand 3-107
saving a file
  using SAVE 3-107
  using SET AUTOSAVE 3-114
saving variables 3-82
scale
  displayed when defining mask 3-132
  displaying
    using SCALE (prefix subcommand) 4-13
    using SET SCALE 3-148
    illustration of 3-148
  removing from screen 3-148
SCALE
  option in QUERY 3-91
  option in SET 3-148
  option in TRANSFER 3-185
SCALE prefix subcommand 4-13
SCHANGE macro 3-108
screen
  changes stacked by READ 3-96
  dividing into multiple logical screens 3-149
  reserving a line on 3-146
  scrolling backward 3-6
  scrolling forward 3-48
SCREEN
  option in QUERY 3-91
  option in SET 3-149
  option in TRANSFER 3-186
screen layout
  in input mode 3-53
  in power typing mode 3-80
screen operation simulation 3-176
scrolling
  backward 3-6
  forward 3-48
search direction
  specifying for column-target 3-19
  specifying for target 3-63
search order, for subcommands and macros
  controlling 3-131
selective change 3-108
sequence of subcommands
  assigning to a PF key 3-139
SEQ8
  option in QUERY 3-91
  option in TRANSFER 3-186
SERIAL
  option in QUERY 3-91
  option in SET 3-151
  option in TRANSFER 3-186
serial identification
  removing 3-151
  specifying 3-151
serialization of file
  controlling 3-151
SET = 3-172
SET APL 3-111
SET ARBCHAR 3-112
  use in CHANGE 3-14
  used with COUNT 3-30
SET AUTOSAVE 3-114
SET CASE 3-116
SET CMDLINE 3-117
SET COLPTR 3-118
SET CTLCHAR 3-118.1
SET CURLINE 3-119
  used to change size of input zone 3-53
SET ESCAPE 3-120
  use in input mode 3-54
SET FILLER 3-121
SET FMODE 3-122
SET FNAME 3-123
SET FTYPE 3-124
SET HEX 3-125
SET IMAGE 3-126
  list of subcommands affected by 3-126
SET IMPCMSCP 3-128
SET LINEND 3-129
SET LRECL 3-130
SET MACRO 3-131
SET MASK 3-132
  example of 3-133
SET MSGMODE 3-134
SET NONDISP 3-135
SET PACK 3-138
SET PFn 3-139
SET POINT 3-141
SET PREFIX 3-143
SET RANGE 3-144
SET RECFM 3-145
SET RESERVED 3-146
SET SCREEN 3-148
SET SCALE 3-149
SET SERIAL 3-151
SET SPAN 3-153
SET STAY 3-155

use in CHANGE 3-14
SET STREAM 3-156
effect in CDELETE 3-10
used with column-target 3-19
SET subcommand
list of options 3-110
SET SYNONYM 3-157
example of 3-159
SET TABLINE 3-160
SET TABS 3-161
used by EXPAND 3-42
using with COMPRESS and EXPAND 3-25
SET TERMINAL 3-162
SET TEXT 3-163
SET TOEOF 3-164
SET TRUNC 3-165
use in CHANGE 3-13
SET VARBLANK 3-166
SET VERIFY 3-167
SET WRAP 3-169
SET ZONE 3-170
use in CHANGE 3-14
SHIFT subcommand 3-173
shifting data 3-173
SIDCODE
option in QUERY 3-91
option in TRANSFER 3-186
size, of logical screens
defining 3-149
SIZE
option in QUERY 3-91
option in TRANSFER 3-186
SORT macro 3-174
example of 3-175
sorting 3-174
SOS option
ALARM 3-176
BFIELD 3-176
CLEAR 3-176
DOWN 3-176
LEFT 3-176
LINEADD 3-176
LINEDEL 3-176
NFIELD 3-176
NLINE 3-176
NULLS 3-176
PFla 3-177
POP 3-177
PUSH 3-177
RIGHT 3-177
TABB 3-177
TABCMD 3-177
TABCMDB 3-177
TABCMDFB 3-177
TABF 3-177
UP 3-177
SOS subcommand 3-176
sounding the alarm 3-41
SPAN
option in QUERY 3-91
option in SET 3-153
option in TRANSFER 3-186
span lines
allowing string target to 3-153
split a line
at column number(s) 3-178
at column pointer 3-178
at cursor 3-178
at string(s) 3-178
SPLIT macro 3-178
assigned to PF key 3-178
eexample of 3-179
spool, printer
copying contents of 3-139
STACK macro 3-180
stacking lines 3-180
status
block incomplete 4-4, 4-5, 4-10
pending 4-4, 4-10
status area
displaying BLOCK INCOMPLETE
resetting 3-103
displaying COPY/MOVE PENDING
resetting 3-103
during macro execution 3-96
during prefix subcommands 4-2
in multiple logical screens 3-149
STATUS macro 3-181
STAY
option in QUERY 3-91
option in SET 3-155
option in TRANSFER 3-186
STREAM
option in QUERY 3-91
option in SET 3-156
option in TRANSFER 3-186
string expression
eexamples of specifying column-target as 3-19
eexamples of specifying target as 3-63
format of 3-19, 3-63
specifying column-target as 3-18
specifying target as 3-63
subcommand
assigning to PF key 3-139
entering in input mode
using escape character 3-120
re-executing 3-195
redisplaying
using & 3-194
using ? 3-196
subcommands
controlling search order for 3-131
multiple
assigned to PF key 3-139
entered on command line 3-129
summary, of XEDIT subcommands and macros G-1
symbolic name
assigning
using .xxxx (prefix subcommand) 4-17
using SET POINT 3-141
displaying 3-90, 3-141
transferring 3-185
synonym
abbreviation of
specifying 3-157
controlling search for 3-157
defining for subcommand 3-157
defining for prefix subcommand 3-143
rearranging operands of 3-157
SYNONYM
option in QUERY 3-91
option in SET 3-157
option in TRANSFER 3-186

SYNONYM check, overriding with COMMAND 3-24

tab character
affected by SET IMAGE 3-126, 3-161
inserted by COMPRESS 3-25
tab characters
how handled by FIND 3-43
how handled by FINDUP 3-43
how handled by NFIND 3-75
how handled by NFINDUP 3-76
in input line 3-53
tab key
PF key used as 3-139
tab line
displaying
using SET TABLINE 3-160
using TABL (prefix subcommand) 4-14
tab settings
defining 3-161
used by EXPAND 3-42
TABL prefix subcommand 4-14
TABLELINE
option in QUERY 3-92
option in SET 3-160
option in TRANSFER 3-186
TABS
option in QUERY 3-92
option in SET 3-161
option in TRANSFER 3-186
target
affected by SET IMAGE 3-127
as absolute line number 3-62
as complex string expression 3-65
as line name 3-63
as relative displacement 3-62
as string expression 3-64
description of 3-62
specifying in hexadecimal 3-63, 3-125
use in ALTER 3-4
use in CHANGE 3-13
use in COMPRESS 3-25
use in COPY 3-28
use in COUNT 3-30
use in DELETE 3-37
use in DUPLICAT 3-40
use in EXPAND 3-42
use in HEXTYPE 3-52
use in LOCATE 3-62
use in LOWERCAS 3-66
use in MOVE 3-70
use in PUT 3-84
use in PUTD 3-86
use in SET RANGE 3-144
use in SHIFT 3-173
use in SORT 3-174
use in STACK 3-180
use in TYPE 3-188
use in UPPERCAS 3-191
TARGET
option in QUERY 3-92
option in TRANSFER 3-186
target search
ignoring case in 3-116
respecting case in 3-116

terminal display
used in full screen mode 3-162
used in line mode 3-162
TERMINAL
option in QUERY 3-92
option in SET 3-162
option in TRANSFER 3-186
terminating editing session
using CANCEL 3-8
using FILE 3-43
using QUIT 3-94
TEXT
option in QUERY 3-92
option in SET 3-163
option in TRANSFER 3-186
TEXT keys
allowing use of 3-163
TOF
option in QUERY 3-92
option in TRANSFER 3-186
TOFEOF
option in QUERY 3-92
option in SET 3-164
option in TRANSFER 3-186
top of file, moving line pointer toward
target 3-63
using UP 3-189
TOP OF FILE line
controlling display of 3-164
top of range 3-144
TOP OF RANGE line
controlling display of 3-164
TOP subcommand 3-182
TRANSFER subcommand 3-183
transferring, setting of editing options 3-183
transferring data between files
using GET 3-49
using PUT 3-84
using PUTD 3-86
translating
characters to lowercase 3-66
characters to uppercase 3-191
transmitting
commands to CMS 3-21
commands to CMS/CP automatically 3-128
commands to CP 3-33
TRUNC
option in QUERY 3-92
option in SET 3-165
option in TRANSFER 3-187
truncation column, defining 3-165
TYPE subcommand 3-188
typewriter terminal
controlling display of column pointer on 3-118
using escape character on 3-120
using input mode 3-54
underscore
used in COVERLAY 3-32
used in FIND 3-43
used in FINDUP 3-43
used in NFIND 3-75
used in NFINDUP 3-76
used in OVERLAY 3-77
unprotected QUIT 3-8, 3-94
UP subcommand 3-189

Index  X – 9
example of 3-190

UPDATE
  option in QUERY 3-92
  option in TRANSFER 3-187
update mode 2-1, 2-2
UPPERCAS subcommand 3-191
  example of 3-191
uppercase
  translating characters entered into 3-116
  translating characters in file to 3-191

VARBLANK
  option in QUERY 3-92
  option in SET 3-166
  option in TRANSFER 3-187
variable number of blanks, significance of 3-166
variable packed record format 3-145
variable record format 3-145
variables
  not restored by RESTORE 3-104
  not saved by PRESERVE 3-82
  restored by RESTORE 3-104
  saved by PRESERVE 3-82
verification
  of changed lines 3-14, 3-167
VERIFY
  option in QUERY 3-92
  option in SET 3-167
  option in TRANSFER 3-187
VERSHIFT
  option in QUERY 3-92
  option in TRANSFER 3-187
viewing data
  to the left 3-57
  to the right 3-105

VMFDEOPT macro F-2
VMFOPT macro F-1
VSBASIC file, renumbering 3-100

warning message
  issued by QUIT 3-94
WIDTH
  option in QUERY 3-92
  option in TRANSFER 3-187
  option in XEDIT 2-1
WRAP
  option in QUERY 3-92
  option in SET 3-169
  option in TRANSFER 3-187
wrap around 3-169
wrapping, automatic line 3-167
writing file on disk
  using FILE 3-43
  using SAVE 3-107
  using SET AUTOSAVE 3-114

XEDIT command 2-1
  relation to LOAD 3-59, 3-60
XEDIT subcommand 3-192

zone
  defining 3-167
  effect on CFIRST 3-170
  effect on CHANGE 3-170
  effect on CLAST 3-170
  moving column pointer to beginning of 3-12
  moving column pointer to end of 3-17
ZONE
  option in QUERY 3-93
  option in SET 3-170
  option in TRANSFER 3-187
This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author’s department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Yes  No

• Does the publication meet your needs?  □  □

• Did you find the material:
  Easy to read and understand?  □  □
  Organized for convenient use?  □  □
  Complete?  □  □
  Well illustrated?  □  □
  Written for your technical level?  □  □

• What is your occupation?
  ____________________________________________

• How do you use this publication:
  As an introduction to the subject?  □  As an instructor in class?  □
  For advanced knowledge of the subject?  □  As a student in class?  □
  To learn about operating procedures?  □  As a reference manual?  □

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)
Reader's Comment Form

If you would like a reply, please print:

Your Name ____________________________ Department ________
Company Name ________________________
Street Address _________________________
City _________________________ State ________ Zip Code ________
IBM Branch Office serving you ________________

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760
IBM Virtual Machine/System Product: System Product Editor Command and Macro Reference

© IBM Corp. 1980, 1982

This Technical Newsletter applies to Release 2 of the Virtual Machine/System Product, Program Number 5664-167. It provides replacement pages for your publication. These replacement pages remain in effect until specifically altered. Pages to be replaced are:

- title page, notices
- iii, iv
- vii, viii
- 3-21, 3-22
- 3-87 – 3-90
- 3-90.1, 3-90.2 (added)
- 3-95, 3-96
- 3-96.1, 3-96.2 (added)
- 3-109, 3-110
- 3-118.1, 3-118.2 (added)
- 3-147, 3-148
- 3-183, 3-184
- 3-184.1, 3-184.2 (added)
- A-1, A-2
- F-1, F-2
- G-3, G-4
- X-1 – back cover

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

This Technical Newsletter incorporates information on the SET CTLCHAR subcommand and the TAG/NOTAG operand of the READ subcommand. It also includes additional filetypes for which the editor has default settings.

For a detailed list of changes, see page iii.

Note: Please insert this page in your publication to provide a record of changes.