z/VM

XEDIT Commands and Macros Reference

Version 5 Release 1.0

SC24-6131-00

z/VM

XEDIT Commands and Macros Reference

Version 5 Release 1.0

SC24-6131-00

Note!
Before using this information and the product it supports, read the information under “Notices” on page 505.

Acknowledgement

IBM gratefully acknowledges the permission to reprint excerpts from the following:

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

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

First Edition (September 2004)

This edition applies to version 5, release 1, modification 0 of IBM z/VM (product number 5741-A05) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-5973-00.

Contents

About This Book

This is a reference book for IBM® z/VM®. It contains all the command formats, syntax rules, and operand and option descriptions, listed alphabetically, for the XEDIT command and XEDIT subcommands and macros.

Who Should Read This Book

This book is written for those with limited data processing experience as well as for more experienced users.

What You Should Know before Reading This Book

You should be familiar with *z/VM: XEDIT User's Guide*, before you try to use this reference book. *z/VM: XEDIT User's Guide* contains tutorial information on using XEDIT and information on writing XEDIT macros using REXX/VM.

Where to Find More Information

For further information, see the books listed under Bibliography on page 511.

How to Send Your Comments to IBM

IBM welcomes your comments. You can send us comments about this book or other VM documentation using any of the following methods:

- Complete and mail the Readers' Comments form (if one is provided at the back of this book) or send your comments to the following address:

  IBM Corporation
  Department 55JA, Mail Station P384
  2455 South Road
  Poughkeepsie, New York 12601-5400
  U.S.A.

  FAX (United States and Canada): 1-845-432-9405
  FAX (Other Countries): +1 845 432 9405
- Send your comments by electronic mail to one of the following addresses:
  - Internet: mhvrcfs@us.ibm.com
  - IBMLink™ (US customers only): IBMUSM10(MHVRCFS)
- Submit your comments through the VM Feedback page ("Contact z/VM") on the z/VM Web site at www.ibm.com/eserver/zseries/zvm/forms/.

Please provide the following information in your comment or note:

- Title and complete publication number of the book (including the suffix)
- Page number, section title, or topic you are commenting on

If you would like a reply, be sure to include your name, postal or email address, and telephone or FAX number.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Summary of Changes

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

SC24-6131-00, z/VM Version 5 Release 1

This edition supports the general availability of z/VM Version 5 Release 1 (z/VM V5R1).

Enabled for LibraryCenter Advanced Search

This document has been enabled for the following z/VM LibraryCenter advanced searches: commands.

SC24-5973-00, z/VM Version 4 Release 4

The edition supports the general availability of z/VM Version 4 Release 4 (z/VM V4R4).

VM NFS Client

The XEDIT command, and the FILE, GET, LOAD, PUT, PUTD, SAVE, and XEDIT subcommands were updated to include the new messages 1019E and 1020E, and the revision return codes 32 and 55.

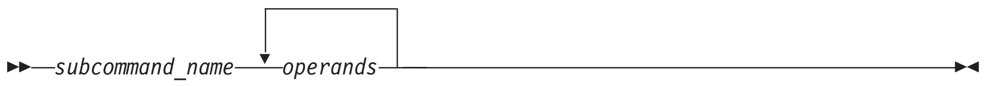
BFS Root Not Case Senstive

The fully qualified VM byte file system root (**/../VMBFS:***filepoolid:filespaceid***/**) is no longer case sensitive.

Chapter 1. Rules and Conventions

XEDIT subcommands and macros both follow the same rules and conventions. In this discussion, *subcommand* refers to both XEDIT subcommands and XEDIT macros.

The general format of XEDIT subcommands is:



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

With more than one operand, separate operands in the command line with at least one blank unless otherwise indicated.

The maximum length of an XEDIT subcommand issued from the XEDIT command line is 255 characters. The maximum length of an XEDIT subcommand issued from an exec or from an XEDIT macro is 256 characters. If either maximum length is exceeded, the editor will ignore the remaining characters, and you will not receive any messages or return codes.

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 the editor performs. For example, the ADD subcommand adds lines to the file.

Subcommand Operands

Subcommand operands are keyword or positional or a combination of both. Operands specify the information on which the editor operates when it performs the subcommand function. Enter the operands in the order which they appear in the command format boxes.

One of the most widely-used operands in XEDIT is the *target* operand, which provides various ways to identify a line to the editor. For more information on the concept of a target, see the LOCATE subcommand in this book and in *z/VM: XEDIT User's Guide*. You should become familiar with targets before attempting to use XEDIT subcommands that require target operands.

Character Set Usage

You enter XEDIT subcommands by using a combination of characters from six different character sets. Table 1 shows each character set and its contents.

Table 1. Character Sets and Their Contents

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

How to Read Syntax Diagrams

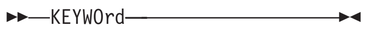
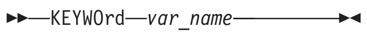
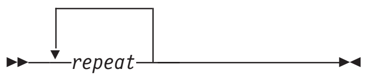
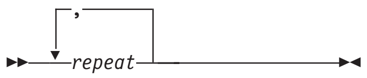
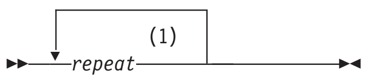
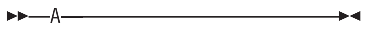
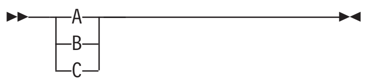
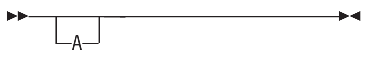
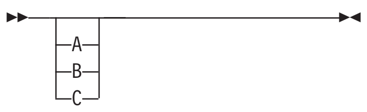
This book uses diagrams to show the syntax of external interfaces and statements. Also, this book uses a special notation to show variable, optional, or alternative content in examples of messages and responses.

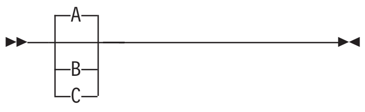
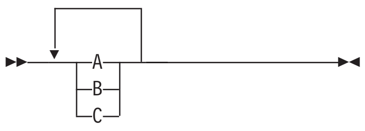
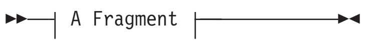
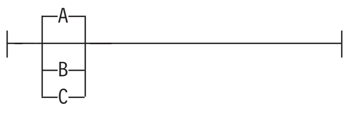
To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►── symbol indicates the beginning of the syntax diagram.
- The ──► symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The ►── symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The ──►◄ symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the following examples.

Syntax Diagram Convention	Example
Keywords and Constants: A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	`►►──KEYWORD──────────────────►◄`

Syntax Diagram Convention	Example
Abbreviations: Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	
Symbols: You must specify these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period
Variables: A variable appears in highlighted lowercase, usually italics. In this example, *var_name* represents a variable that you must specify following KEYWORD.	
Repetitions: An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	   **Notes:** 1 Specify *repeat* up to 5 times.
Required Item or Choice: When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	 
Optional Item or Choice: When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 

Syntax Diagram Convention	Example
Defaults: When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice: A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragment: Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named “A Fragment.”	 **A Fragment:** 

Message and Response Notation

This book may include examples of messages or responses. Although most messages and responses are shown exactly as they would appear, some content may depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

xxx	Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.
[ ]	Brackets enclose optional items that may be displayed.
{ }	Braces enclose alternative items, only one of which will be displayed.
|	The vertical bar separates items within brackets or braces.
...	The ellipsis indicates that the preceding item may be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, may be repeated.

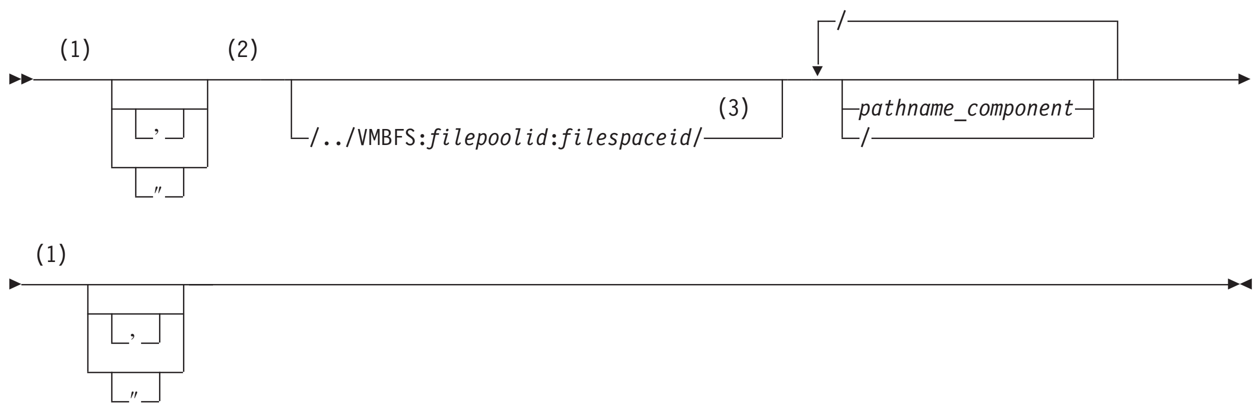
Understanding Byte File System (BFS) Path Name Syntax

All OpenExtensions Byte File System (BFS) objects (files, directories, and so on) are identified through path names. A path name has an optional beginning slash, followed by one or more path name components separated by slashes. A path name component is a string of characters used to identify a BFS object.

A BFS path name may represent a file system accessed through the Network File System (NFS). The NFS file system may be on a remote or local system, which may be VM or non-VM. The OPENVM MOUNT command or the mount (BPX1MNT) callable service links an NFS file system to a BFS path name, enabling it to be used on most commands and interfaces that accept BFS path names.

The BFS path name identifier is shown as one word, *pathname* when it depicts a specific path name variable.

Format



Notes:

1 A single quote or a double quote is an optional delimiter. If specified to signify the start of a path name, the identical delimiter must also be specified to signify the end of the path name. (See usage note 7 on page 8.)

2 The minimum path name is a single slash (/).

3 The ending slash is required only if one or more path name components are also specified.

Operands

/../VMBFS
is a keyword string that indicates a **fully qualified VM byte file system root**, which identifies the byte file system in which the specified object resides. The VMBFS keyword is not case sensitive.

: (colon)
is a separator that must be specified following the VMBFS keyword and the *filepoolid*.

filepoolid
is the name of the file pool that contains the byte file system. The file pool name can be up to eight characters long. The first character must be alphabetic, but the remaining characters can be alphabetic or numeric. This name is not case sensitive.

filespaceid
is the name of the file space in which the byte file system resides. The file space ID can be up to eight characters long. This name is not case sensitive.

/ (slash)
is a separator that must be specified after the *filespaceid* if path name components are specified. The slash must also be specified between path name components.

When **/** is specified as a single-character path name, it indicates the root (top) directory of the currently-mounted byte file system.

Note: A path name must not start with two slashes when in the XEDIT environment.

pathname_component
is the name of an object in the BFS hierarchy. A path name component may be 1-255 characters in length. The slash character (/) and the null character (X'00') are not valid within a path name component. If multiple path name components are specified, they must be separated by slashes. All path name components prior to the last one specified are interpreted as directory names in the hierarchy. The last path name component, if not followed by a slash, may or may not be a directory. When the last path name component is followed by a slash, it is always interpreted as a directory.

Path name component names are case sensitive.

Usage Notes

1. A byte file system may be enrolled in the same file pool as other byte file systems and SFS users.
2. In the OpenExtensions environment, all byte file systems are uniquely identified with the **/../vmbfs:***filepoolid***:***filespaceid* construct.
3. Path names can be specified in several ways:
   - When the first character of the path name is not a slash, the path name is known as a **relative path name**. The search for the file starts at the working directory. To establish the working directory, use the OPENVM SET DIRECTORY command or the chdir (BPX1CHD) callable service. To find the value of the current working directory, use the OPENVM QUERY DIRECTORY command or the getcwd (BPX1GCW) callable service.
   - When **/../vmbfs:***filepoolid***:***filespaceid* is specified at the start of a path name, it is referred to as a **fully qualified path name**. The file is searched for in the byte file system, which is defined as file space *filespaceid* in file pool *filepoolid*. The byte file system does not need to be explicitly mounted.
   - When the path name starts with a slash (but not **/../vmbfs:***filepoolid***:***filespaceid*), the path name is known as an **absolute path name**. The search for the file starts from the root of the currently mounted byte file system. The root directory can be established by using the OPENVM MOUNT command or the mount (BPX1MNT) callable service, or by the POSIXINFO FSROOT statement in your CP directory entry. To find the value of the root directory, use the OPENVM QUERY MOUNT command or the uname (BPX1UNA) callable service.

   Refer to *z/VM: OpenExtensions Commands Reference* for more information on the OPENVM commands. Refer to *z/VM: OpenExtensions Callable Services Reference* for more information on the callable services.
4. The entire path name must be in the range of 1-1023 characters. Individual path name components cannot exceed 255 characters. All characters are valid within a path name, with the following restrictions:
   - The null character (X'00') is not permitted within a path name.
   - A slash (/) is interpreted as the delineator of a path name component.

For an application to be portable to the broadest set of environments, POSIX standards suggest the application restrict the path name as follows:

- Do not exceed 14 characters for any path name component.
- Use only these characters:

A-Z	Uppercase alphabetic
a-z	Lowercase alphabetic
0-9	Numeric
.	Period
_	Underscore
-	Dash

5. Path name components are case sensitive. Note that Abc, abC, and ABC are valid unique path name components. When a path name is entered on the CMS command line, it will not be uppercased. However, a path name entered on the XEDIT command line will be uppercased when SET CASE UPPER is in effect.
6. There are two path name components (file names) that have special meaning during path name resolution. These are:

. The path name component consisting of the single dot character (**.**). When dot is encountered in the path name, it refers to the directory specified by the preceding path name component.

Some dot (.) examples:

a. If you specified a path name of:

```
/joes/recipes/./pie
```

It would be equivalent to:

```
/joes/recipes/pie
```

b. If you specified a path name of:

```
./joes
```

It would be equivalent to:

```
joes
```

.. The path name component consisting of two dot (**..**) characters. When dot dot is encountered in the path name, it refers to the parent directory of its predecessor. As a special case, in the root directory, dot dot refers to the root directory itself. The construct **/../vmbfs:***filepoolid***:***filespaceid*, as described above, is the only exception.

Some dot dot (..) examples:

a. If you had previously set your working directory (using OPENVM SET DIRECTORY) to:

```
/joes/recipes/
```

And you specified a relative path name of `../tools`, this would be equivalent to specifying an absolute path name of:

```
/joes/tools
```

b. If you are working in `/bin/util/src`, and you want to go to `/bin/util`, you can enter:

```
openvm set directory ..
```

   c. If you are working in `/u/rexx/prog/src`, and you want to refer to the file `test` in the directory `/u/rexx/appl/examples`, you could use the following path name to refer to that file:

      ```
      ../../appl/examples/test
      ```

7. Enclose a BFS path name within single quotes ('*pathname*') or double quotes ("*pathname*") if it contains any of the following characters. Results are unpredictable if a path name or path name component contains any of these characters and it is not enclosed within quotes.

	Blank space
(	Left parenthesis
)	Right parenthesis
'	Single quote
"	Double quote
*****	Asterisk
=	Equal sign
X'FF'	Hexadecimal FF

   Notes:

   a. If a path name includes a single quote, specify the path name in one of these ways:
      - Place double quotes around the path name.
      - Place single quotes around the path name, but be sure to use two additional single quotes to denote the single quote that is part of the path name.

   b. If a path name includes a double quote, specify the path name in one of these ways:
      - Place single quotes around the path name.
      - Place double quotes around the path name, but be sure to use two double quotes to denote the double quote that is part of the path name.

   c. All characters are taken literally; no symbolic substitution is done.

   Some Examples:

   a. To XEDIT a file called `my dir/my file` that is directly under your root directory, you may specify:

      ```
      xedit '/my dir/my file' (nametype bfs
      ```

      The NAMETYPE BFS option was specified to distinguish the file being edited as a BFS file instead of a CMS file.

      Note that:

      ```
      xedit /a/b/c
      ```

      is equivalent to:

      ```
      xedit '/a/b/c'
      ```

   b. To edit a file called `/a/b b'/c`, you can enter the name in either of the following ways:

      ```
      xedit '/a/b b''/c'
      xedit "/a/b b'/c"
      ```

   c. To edit a file called `/a/b b"/c`, you can enter the name in either of the following ways:

```
xedit "/a/b b""/c"
xedit '/a/b b"/c'
```

8. Multiple adjacent slashes (//) in a path name are interpreted as a single slash by XEDIT. However, these multiple slashes are included in the maximum path name length check.

   Note: A path name must not start with two slashes in the XEDIT environment.
9. **If you are using a # character in your BFS path name,** you may need to change your terminal settings in order to specify a path name that contains certain special characters. For example, you want to use a # in a name, but the default line end character is a #. So you may have to change your logical line end symbol using the CP TERMINAL LINEND command.

   If you wish to enclose a BFS path name containing blanks in double quotes, you may need to use the CP TERMINAL ESCAPE command to change your logical escape symbol because the default value is a double quote. Use the CP QUERY TERMINAL command to display the special characters in effect for your terminal.

   Use QUERY LINEND and SET LINEND to find out and define your current linend character for full-screen CMS.

Using the Online HELP Facility

You can receive online information about the commands described in this book using the z/VM HELP Facility. For example, to display a menu of XEDIT commands, enter:

```
help xedit menu
```

To display information about a specific XEDIT command (ADD in this example), enter:

```
help xedit add
```

To display information about the XEDIT command enter:

```
help xedit
```

You can also display information about a message by entering one of the following commands:

help *msgid* **or** help msg *msgid*

For example, to display information about message DMS001E, you can enter one of the following commands:

help DMS001E **or** help msg DMS001E

For more information about using the HELP Facility, see *z/VM: CMS User's Guide*. To display the main HELP Task Menu, enter:

```
help
```

For more information about the HELP command, see *z/VM: CMS Commands and Utilities Reference* or enter:

```
help cms help
```

Chapter 2. The XEDIT Command

```
                                            (2)
          (1)                  ┌─*──┐    ┌─(─────────────────────────────────────────────┐
►►──Xedit──────┬─fn──ft─┼────┼─┬──┼───────────────────────────────────────────────┴──────────►◄
               │        └─fm─┘ │  │  (3)
               └─pathname──────┘  └─(───────┤ Options ├─┤ Update Mode Options ├─┬───┬─┘
                                                                              └─)─┘
```

Options:

```
   ┌─WINdow──XEDIT─┐                                  ┌─PROFile──PROFILE────┐
├──┼───────────────┼──┬─────────────┬──┬───────────┬──┼─────────────────────┼──┬─────────┬──►
   └─WINdow──wname─┘  └─Width──width┘  └─NOSCreen──┘  ├─PROFile──macroname──┤  └─NOCLear─┘
                                                      └─NOPROFil────────────┘

                               ┌─LOCk───┐  ┌─NOUpdate─┐  ┌─BFSLine──NL──────────────┐
►──┬───────┬──┬─────────────────────┬──┼────────┼──┼──────────┼──┼──────────────────────────┼──►
   └─NOMsg─┘  └─MEMber──membername──┘  └─NOLOCk─┘  └─Update───┘  └─BFSLine──┬─lrecl──────┬─┘
                                                                           ├─CRLF───────┤
                                                                           ├─CRNL───────┤
                                                                           ├─/string/───┤
                                                                           └─/hexstring/┘

   ┌─NAMetype──CMS─┐
►──┼───────────────┼────────────────────────────────────────────────────────────────────────┤
   └─NAMetype──BFS─┘
```

Update Mode Options:

```
   ┌─Seq8───┐                                              ┌─Incr──1────┐
├──┼────────┼──┬───────────┬──┬───────┬──┬─────────────┬──┼────────────┼──┬─────────────────┬──┤
   └─NOSeq8─┘  ├─Ctl──fn1──┤  └─Merge─┘  └─UNtil──ft1──┘  └─Incr──incr─┘  └─SIDcode──string─┘
               └─NOCtl─────┘
```

Notes:

1 If a profile is invoked which issues a LOAD subcommand that specifies *fn* and *ft* or *pathname*, *fn* and *ft* or *pathname* are not required.

2 The default options are shown above the main line in the options groups.

3 You can enter options in any order between the parentheses. If a default is not shown for an option, refer to the option description for the information.

Purpose

Use the CMS command XEDIT to invoke the editor to create, modify, and manipulate CMS files on disk, SFS files, or BFS regular files in the byte file system. Once you invoke the editor, you can execute XEDIT subcommands and use REXX or the EXEC 2 macro facility.

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

Operands

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

  In the GUI environment, if *New file* is selected from a CMSDESK application, and Xedit has been selected as your editor preference, the XEDIT file ID input window is displayed for CMS files.

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

- Editing existing files

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

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

pathname
: is the name of the BFS file to be edited. See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name syntax.

Options

WINdow *wname*
: is the name of the virtual screen and window XEDIT uses to display all files being edited in the ring. By default, XEDIT uses the window and virtual screen named "XEDIT". This option is ignored in a GUI environment.

  Note: The window name must not contain invalid file name characters. Also, *CMS* or *WM* cannot be used as the window name.

  For more information on virtual screens and windows, see Appendix G, "XEDIT Virtual Screens and Windows," on page 487.

Width *width*
: defines the amount of virtual storage containing 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. For SFS and minidisk files, if WIDTH is not specified in either the XEDIT command or the LOAD subcommand, the default is the larger of the following:

  - Logical record length (LRECL) of the file.
  - Default logical record length associated with the file type. See page 453 for a list of these defaults.

  For BFS files, if WIDTH is not specified in either the XEDIT command or the LOAD subcommand, the default depends upon the BFSLINE value.

  - For BFSLINE *lrecl*, the default is *lrecl*.

- Otherwise, the default value is 80 or the length of the longest line, whichever is larger.

NOSCreen
forces a 3270 display terminal into line (typewriter) mode. This option is ignored in a GUI environment.

PROFile *macroname*
If the specified macro exists on one of the accessed minidisks or SFS directories, the editor executes it as the first subcommand. If the specified macro exists and is empty, an error message is displayed.

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

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

The profile macro must always have a file type of XEDIT. The profile macro may not reside in the byte file system.

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

NOCLear
specifies 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 option is ignored for the GUI window, it only affects an emulator session.

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

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

CMS supports SFS MACLIBs which do not have members; however, CMS does not support empty members. If the MACLIB file is empty, an error message is displayed.

MEMBER is ignored if you attempt to use it when editing a BFS file.

LOCk
causes the editor to lock an existing SFS file to prevent other users from modifying the file while you are editing it, or causes the editor to obtain an advisory lock for an existing BFS file while it is being read into the XEDIT session. Note that a BFS file does not remain locked while in the session, and other users may choose to override an advisory lock.

The LOCK option is ignored for files on minidisks and files that reside in SFS directory control directories and files in NFS-mounted directories.

For files in SFS file control directories, you must have write authority to the file to lock it. If you have only read authority, a warning is displayed and the editing session continues without locking the file. LOCK is the default.

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

NOLOCk

indicates you do not want the editor to try to lock the file.

For SFS files:

You can use this option to edit a file another user has locked SHARE or UPDATE. If you specify NOLOCK, other users can change the file while you are editing it. NOLOCK is the default for files that reside in SFS directory control directories.

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

For BFS files:

The editor does not obtain an advisory lock while reading the file into the XEDIT session.

NOUpdate

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

Update

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

If the file does not exist, the editor creates a new UPDATE file with a file mode of A1 to contain modifications the user makes. If the original source file (base file) is empty, an error message is displayed. If an update file is empty, an informational message is displayed. For more information on the XEDIT UPDATE option, see *z/VM: CMS Application Development Guide*.

UPDATE is ignored if you attempt to use it when editing a BFS file.

BFSLine

Use the BFSLINE option to tell XEDIT how to translate a BFS byte stream into records when reading the file into virtual storage for editing and to set the initial setting of SET BFSLINE.

You can define an end-of-line character or characters for use in interpreting lines in a BFS file if you specify anything other than BFSLINE *lrecl*. When a file is read into an XEDIT session, everything up to the end-of-line character is interpreted as a line and presented as a ‘record’ in the XEDIT session. The end-of-line character is not displayed while XEDITing the BFS file. When a FILE, PUT, PUTD, or SAVE subcommand is entered for the file when it is being written back to a byte file system, the setting of SET BFSLINE is used to determine the end-of-line character that will be used. If the setting has not been changed from its initial value, the same end-of-line character that existed previously will be used. See the description for SET BFSLINE in “SET BFSLINE” on page 245 for more information.

BFSLINE *lrecl* does not separate the file into records based on an end-of-line character.

The BFSLINE option determines the initial setting used for record format (RECFM). For BFSLINE *lrecl*, this initial setting is fixed (F). For any other BFSLINE value, it is variable (V). Changing this value while in an XEDIT session will affect the way the file is stored in the byte file system. See the SET RECFM command description in “SET RECFM” on page 319 for more information.

If not specified, BFSLINE NL is the default.

NL indicates the new line character (X'15') should be used to delineate lines when reading or writing a BFS file.

lrecl indicates the file should be treated as a fixed file, with no interpretation of records based on end-of-line characters. When BFSLINE *lrecl* is in effect, the file is presented as a fixed record format (RECFM=F) file with a logical record length (LRECL) equal to the *lrecl* value. No end-of-line characters are removed from or added to the file when it is read from or written back out to the byte file system if BFSLINE *lrecl* is still in effect.

The last record will be padded with blanks if *lrecl* is something greater than 1 and the last record does not completely fill the last logical record.

CRLF indicates the carriage return and line feed characters (X'0D25') should be used to delineate lines when reading or writing a BFS file.

CRNL indicates the carriage return and new line characters (X'0D15') should be used to delineate lines when reading or writing a BFS file.

/string/ allows the user to specify 1 - 2 characters that are used to delineate lines when reading or writing a BFS file. These characters are translated to uppercase before they are utilized. The slash character (/) may not be one of these characters (in other words, you cannot enter ///).

/hexstring/
specifies a hexadecimal string of 2 or 4 characters that defines the value to be used for BFSLINE. The *hexstring* must be in the format X'nnnn' or X'nn'. You must not specify any spaces in the string, and there must be 2 or 4 hexadecimal characters in the string.

BFSLINE is not used unless NAMETYPE BFS is in effect.

NAMetype
Use the NAMETYPE option to specify whether the file ID will be interpreted to be in the form used for CMS record files or in the form used for byte file system (BFS) files.

CMS Indicates file IDs will be interpreted to be CMS record files. They will be interpreted as *fn ft fm*. This is the default.

BFS Indicates file IDs will be interpreted as path names (*pathname*). If you are in a GUI environment and NAMETYPE BFS has been specified, the XEDIT file ID prompt is not displayed. Refer to “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the BFS path name syntax.

The following options are significant only if XEDIT is to be used in update mode, but they are ignored if NAMETYPE BFS is specified:

Seq8
specifies the entire sequence field (the last eight columns of each file line)

contains an eight-digit sequence number. The SEQ8 option automatically forces the UPDATE option. SEQ8 is the default value.

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

The NOSEQ8 option forces the UPDATE option.

Ctl *fn1*
specifies *fn1* CNTRL is an update control file that controls the application of multiple update files to the file to be edited. If the control file is empty, an error message is displayed. (For more information on the CMS UPDATE command, see *z/VM: CMS Commands and Utilities Reference*).

This option automatically forces the UPDATE and SEQ8 options.

A maximum of 32 unique AUX file names may be specified in *fn1* CNTRL. If an auxiliary control file is empty, an informational message is displayed.

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

Merge
specifies 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.

Note: The update file resulting from the merge may have different sequence numbers than if the updates were applied individually.

UNtil *ft1*
specifies the file type of the last update to be applied to the file. Changes are applied to the file being edited from all file types in the control file, up to and including the file type specified with the UNTIL option.

With the UNTIL option, you can specify file types of update files listed in the control file or of update files listed in an auxiliary control file. Do not specify AUX file types (AUX*xxxxx*) with the UNTIL option.

The UNTIL option forces the UPDATE option.

Incr *incr*
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 1. This option forces the UPDATE option.

SIDcode *string*
specifies a string the editor inserts in every line of an update file, whether the update file is being created or is an existing file. The editor inserts the specified string in the first eight columns of the last 17 columns of the file line (lrecl-16 to lrecl-9). For example, if you have a file with fixed, 80-character lines, the editor inserts the string in columns 64 through 71. If the string is fewer than eight characters, it is padded on the right with blanks. Any data in the eight columns is overlaid.

This option forces the UPDATE option.

Usage Notes

1. To use XEDIT on a file in an SFS directory, the directory must be accessed and you must have authorization to access the file. For a file in an SFS file control directory, you must have either read or write authority. For files in SFS directory control directories, you must have either directory read or directory write authority for the directory in which the file resides. For more information on SFS directories and authorizations, see *z/VM: CMS User's Guide*.

   To use XEDIT on a BFS file, you must have permission to access the file. Refer to the OPENVM PERMIT command description in *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM PERMIT for more information.
2. When the LOCK option is in effect for an SFS file, it is possible for the lock to be removed during your edit session if one of the following abnormal errors occurs:
   - file pool server failure
   - network or Advanced Program-to-Program Communications/VM (APPC/VM) failure on the last data link with the file pool server
   - your virtual machine abends or is re-IPLed

   Note: If an abend occurs, the update session lock obtained by XEDIT will be deleted if sufficient storage is available for this additional processing.
3. To change files in an SFS directory control directory, you must access the directory in read/write mode. Because CMS lets only one user at a time access an SFS directory control directory in read/write mode, no one can change a file while you are editing it.

   You can use XEDIT on files within a directory control directory that is accessed in read-only mode. No warning message is displayed, but because the directory is accessed in read-only mode, you cannot change the copy of the file in the directory by using XEDIT subcommands such as FILE and SAVE. To save the changes, you must save or file the file on a minidisk or directory you have accessed in read/write mode.

   To change files in a file control directory, you can access the directory in either read-only or read-write mode. (By default, when you access someone else's directory, the access mode is read-only.) To have XEDIT respect the read-only access for file control directories regardless of your file authority, use the CMS SET RORESPECT ON command. (See *z/VM: CMS Commands and Utilities Reference* for an explanation of this command.)
4. The CMS RORESPECT setting may be used to prevent XEDIT from writing to a file control directory accessed read-only. To prevent XEDIT from locking a file in the directory so others may write to it, you may query the CMS setting and issue a LOAD subcommand with the NOLOCK option from your profile.
5. If you XEDIT an SFS or BFS file that has been migrated (moved to DFSMS/VM-owned storage), it is automatically recalled if the CMS SET RECALL command is set to ON. (See *z/VM: CMS Commands and Utilities Reference* for an explanation of this command.) This may cause a slight delay before the file can be accessed. If CMS SET RECALL is OFF the file will not be recalled and an error message will be displayed.
6. For the PROFILE, CTL, SIDCODE, INCR, UNTIL, MEMBER, WIDTH, WINDOW, BFSLINE, and NAMETYPE options, the operand must be specified; otherwise, the next option is interpreted as its operand. For example, in the PROFILE *macroname* option, *macroname* must be specified; if it is not, the next option is interpreted as the operand *macroname*.

7. Once the XEDIT *command* has been executed, the XEDIT *subcommand* can be used to edit and display multiple files simultaneously (see “XEDIT” on page 410).
8. You can also call the editor recursively (that is, using the CMS XEDIT command). This ability is particularly useful when applications are developed using the editor and its macro facilities to interface with the user, for example, HELP.

   When you call CMS XEDIT recursively, a new ring of files is begun that is independent of any previous ring(s).
9. The MEMBER option and the NOUPDATE option have no effect when preceded by an option that automatically forces update processing. Also, options that usually force update processing are ignored when the MEMBER option or the NOUPDATE option precedes them.
10. If full-screen CMS is set to ON before XEDIT writes to the screen, XEDIT issues the WINDOW SHOW CMSOUT command followed by WINDOW SHOW XEDIT or a WINDOW SHOW for the particular window that has been set up to display the file.
11. The editor is kept in virtual storage as part of the CMS nucleus shared segment; the CMS user area is unused. As a result, assuming a large enough virtual machine, any CMS or CP command may be issued directly from the editor environment itself (if a SET IMPCMSCP subcommand is in effect).
12. When an XEDIT command invokes the PROFILE macro, everything following the command name XEDIT is tokenized (truncated to eight characters), and then assigned to the argument string passed to the PROFILE macro as the first parameter.

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

    When you specify the NAMETYPE BFS option on the XEDIT command, your profile is called as a REXX function, and the untokenized, mixed case file ID (shown below as *fileid2*) is passed unchanged as a second argument string.

    Note: The second parameter (*fileid2*) is still enclosed in quotes if that is the way it was entered.

    This is useful when editing a BFS file because the path name may be up to 1023 bytes long, is case sensitive, and may contain blanks and other special characters, such as quotes. The XEDIT profile must be in REXX, not in EXEC or EXEC2, when the NAMETYPE BFS option is used. The return code is set to **RC=5** if the profile returns an invalid result.

    Whether or not NAMETYPE BFS is specified, special substitution is done for a file ID enclosed in quotes. It is assumed this is a BFS path name containing special characters such as blanks or a (, and a place holder, the character ‘*’, is substituted in its place in the argument list for the first argument string so the profile option processing works correctly.

    An example of a way to obtain these argument strings from within your profile when the NAMETYPE BFS option is used is:

    ```
    PARSE ARG fileid1 ft fm '(' OPTIONS , fileid2
    ```

    In the example above, *fileid1* can be :
    - The file name for a CMS file
    - A path name for a BFS file that does not contain any blanks or special characters
    - A placeholder for a file ID enclosed in quotes on the command invocation

13. When you enter an XEDIT command for an SFS or minidisk variable-format file, trailing blanks are removed when it is filed (or saved).

    When the record format is variable (V) for a BFS file, trailing blanks are removed when it is filed (or saved), and a line containing all blanks is represented in the file as two BFSLINE values in a row. (The initial record format setting for a BFS file depends upon the BFSLINE option. See page 14 for the BFSLINE option description on the XEDIT command for more information.)
14. Update control comment records cannot be edited when the UPDATE option is in effect. Edit the update file without specifying any update options to change these records.
15. Many languages have more characters than can be displayed using one-byte codes (KANJI, for example). A Double-byte Character Set (DBCS) represents these characters. The double-byte characters can appear in a sentence with characters from other languages displayed in 1-byte codes. Files containing double-byte characters are handled differently from files that contain only 1-byte characters.

    When specifying a BFSLINE value for use on files containing DBCS characters, ensure you use a value that will not conflict with DBCS characters. The hexadecimal code for a DBCS character must be X'0000', X'4040', or X'*aabb*', with *aa* and *bb* both being in the range of X'41' to X'FE'.
16. The virtual storage required to edit a ring of one or more files is greater than the size of the files themselves. This includes storage necessary to manage the ring, the lines, the display, and other internal XEDIT storage needs.
17. Unless BFSLINE *lrecl* is in effect when you XEDIT a BFS file, a blank line is inserted if two BFSLINE values are found in a row.
18. The BFSLINE setting has no effect when a PUT, PUTD, GET, FILE, SAVE, or LOAD subcommand is used to read or write a CMS record file when NAMETYPE CMS is in effect.
19. The BFSLINE option can also be used on the XEDIT subcommand from within an XEDIT session. For example, a user can enter:

    ```
    XEDIT a a a
    ```

    and then enter

    ```
    XEDIT pathname (BFSLINE NL NAMETYPE BFS
    ```

    on the XEDIT command line.
20. Commit behavior of BFS files is different from that of SFS or minidisk files. For example, if the editor encounters an error when writing to a minidisk or SFS file, your changes are reversed and the original file is preserved. This is not the case for BFS files. The editor creates a copy of the file in XEDTEMP CMSUT1 on your A disk so you can recover your data if an error occurs when writing the file to the byte file system.

    If the editor is unable to create the XEDTEMP CMSUT1 file, you will receive a message describing the problem, and a second message:

    ```
    595E  Not able to create CMS file used for recovery.  Correct
          error or QQUIT to exit without writing file
    ```

    If the editor encounters an error while writing the byte file system file such that the contents of an existing file are damaged, you will receive a message describing the problem, and a second message to tell you that you must take action to recover your file.

    ```
    024E  File XEDTEMP CMSUT1 A1 contains file contents; use
          OPENVM PUT to recover file
    ```

21. When a new BFS file is created, the owning UID established is the effective UID of the VM User ID on which the request was issued. The GID is the GID of the parent directory. Use OPENVM OWNER to change the defaults. Refer to *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM OWNER for more information.
22. Permissions for a new BFS file are set based on the current value of the mask, with the exception of execute permissions, which are not set to ON. Use OPENVM PERMIT to change the defaults. See *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM for more information on these OPENVM commands.
23. You can XEDIT path names only if they represent BFS regular files.
24. When you are in a GUI environment, initially the GUI windows for rings are arbitrarily positioned on the workstation screen. You cannot control the initial position. However, you can move the windows once they are displayed using workstation controls such as a pointing device or keys.

    The initial size of your GUI XEDIT window is determined by your terminal emulator session setting. The window can be resized to a smaller size by using workstation controls to move the borders.

    When using XEDIT to edit a file in a GUI environment, the title bar of the GUI window contains the word "XEDIT".

    Note: It is recommended the workstation agent be configured to use a non-proportional font. If you elect to use proportional fonts in your xedit session, columns and rows may not appear to line up.
25. Typically you would use the default BFSLINE option when XEDITing a file in an NFS-mounted file system.

    If the NFS-mounted file system is a minidisk or SFS directory, you must coordinate XEDIT's BFSLINE option with the VMNFS server options specified on the mount. This is necessary because the NFS-mounted file is changed into byte-stream format when read by XEDIT, and changed back into record format when written to the minidisk or SFS directory through NFS.

    If the remote CMS file is a Variable file, you could use *lines=NL* on the mount and the default **BFSLINES NL** in the XEDIT options. If the remote CMS file is a Fixed file, you could specify *lines=none* in the mount options and **BFSLINES** *lrecl* in the XEDIT options.

Responses

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

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

The editing session continues but the file is not locked.

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

```
178I  Updating fn ft fm
      Applying fn ft fm
1229I fn ft fm is empty
            .
            .
            .
180W Missing PTF file fn ft fm
```

If the XEDIT work file, XEDTEMP CMSUT1, exists on a file mode accessed R/W as a result of a previous edit session that ended abnormally, you will receive the message:

```
024E   File XEDTEMP CMSUT1 fm already exists
```

You can use the CMS TYPE command to examine the existing file. If you decide you want to keep it, use the CMS RENAME command to give it a new CMS file ID, or use OPENVM PUTBFS to move it into the byte file system. Refer to *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM PUTBFS for more information on this command. If the file is incorrect or incomplete, erase it and enter the command again.

If you try to XEDIT an existing empty file and a profile is **not** executed, you get the following message:

```
556I  Editing existing empty file:
```

If you are in a GUI environment and *fn ft* are not specified for a CMS file, the XEDIT File ID window is displayed.

Messages and Return Codes

002E	**File** *fn ft fm* **not found [RC=28]**
003E	**Invalid option:** *option* **[RC=24]**
024E	**File XEDTEMP CMSUT1** *fm* **already exists [RC=28]**
029E	**Invalid parameter** *parameter* **in the option** *option* **field [RC=24]**
033E	**File is not a regular BFS file [RC=32]**
048E	**Invalid filemode** *mode* **[RC=24]**
054E	**Incomplete [or incorrect] fileid specified [RC=24]**
062E	**Invalid character in fileid {***fn ft fm***|***pathname***} [RC=20]**
065E	*option* **option specified twice [RC=24]**
066E	*option1* **and** *option2* **are conflicting options [RC=24]**
069E	**Filemode** *mode* **not accessed [RC=36]**
070E	**Invalid parameter** *parameter* **[RC=24]**
104S	**Error** *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, or 100]**
109S	**Virtual storage capacity exceeded [RC=104]**
132S	**File [***fn ft fm***] too large[:** *pathname***] [RC=88]**
137S	**Error** *nn* **on STATE for** *fn ft fm* **[RC=88]**
229E	**Unsupported OS dataset, error** *nn* **[RC=80, 81, 82, or 83]**
500E	**Unable to unpack file** *fn ft fm* **[RC=88]**
508E	**LOAD must be the first subcommand in the profile [RC=3]**
512E	**This is not allowed in CMS subset mode [RC=100]**
554E	**Not enough virtual storage available [RC=104]**
556I	**Editing existing empty file:**
571I	**Creating new file:**
622E	**Insufficient free storage (for {MSGLINE|PFkey/PAkey|synonyms})**
915E	**Maximum number of windows already defined [RC=13]**
927E	**The virtual screen must contain at least 5 lines and 20 columns [RC=24]**
928E	**Command is not valid for virtual screen CMS [RC=12]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code** *rc* **and reason code** *rs* **[RC=55, 104]**
1138E	**File sharing conflict for file {***fn ft fm***|***pathname***] [RC=70]**
1214W	**File** *fn ft fm* **already locked SHARE**
1215E	**File** *fn ft fm* **is locked or in use by another user [RC=70]**

1229E	*fn ft fm* **is empty [RC=88]**
1229I	*fn ft fm* **is empty**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=31, 55, 70, 76, 99, or 100]**
1299W	**Warning: Not authorized to lock file** *fn ft fm*
1300E	**Error** *nn* **{locking|unlocking} file** *fn ft {fm|dirname}* **[RC=55, 70, 76, 99, or 100]**
2105E	**Permission is denied [24]**
2134E	**Return code** *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname*] **[RC=100]**
2154E	**File {***fn ft fm***|***pathname***} is migrated and implicit RECALL is set to OFF [RC=50]**
2155E	**DFSMS/VM error occurred during creation or recall of file {***fn ft fm***|***pathname***} [RC=51]**
2526E	**File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code** *acs rcode* **[RC=51]**

Messages with Member Options

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

Messages with Update Option

007E	**File** *fn ft fm* **is not fixed, 80-character records [RC=32]**
007E	**File** *fn ft fm* **does not have a logical record length greater than or equal to 80 [RC=32]**
007E	**File** *fn ft fm* **does not have the same format and record length as** *fn ft fm* **[RC=32]**
007E	**File** *fn ft fm* **is not fixed record format [RC=32]**
104S	**Error** *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 32, or 55]**
174W	**Sequence error introduced in output file:** *seqno1* **to** *seqno2* **[RC=32]**
178I	**Applying** *fn ft fm*
179E	**Missing or invalid MACS card in control file** *fn ft fm*
180W	**Missing PTF file** *fn ft fm*
183E	**Invalid {CONTROL|AUX} file control card [RC=32]**
184W	**./ S not first card in update file--ignored [RC=32]**
185W	**Non numeric character in sequence field** *seqno* **[RC=32]**
186W	**Sequence number not found [RC=32]**
207W	**Invalid update file control card [RC=32]**
210W	**Input file sequence error:** *seqno1* **to** *seqno2* **[RC=32]**
317E	**Number of AUX file types in control file** *fn ft fm* **exceeds 32 [RC=32]**
570W	**Update** *ft* **specified in the UNTIL option field not found**
597E	**Unable to merge updates containing ./ S cards [RC=32]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=31, 55, 70, 76, 99, or 100]**

where return codes are:

0	Normal
3	LOAD must be the first subcommand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Command is not valid for virtual screen
13	Maximum number of windows already defined
20	A character in the file name, file type, or path name is not valid.

24	Invalid parameters or options
28	Source file not found (UPDATE MODE), or library not found (MEMBER option), or specified PROFILE macro does not exist, or file XEDTEMP CMSUT1 already exists
31	A rollback occurred
32	Error during updating process, or file is not a library, or library has no entries, or file is not fixed, 80 char. records, or maximum number of AUX file types exceeded, or BFS file is not a regular file, or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
88	File is too large and does not fit into storage, a previous MACLIB function was not finished, or an unsupported function was attempted with an empty file
99	A required system resource is not available
100	Error reading the file into storage
104	Insufficient storage available

Chapter 3. XEDIT Subcommands and Macros

This chapter describes the formats and operands of the XEDIT subcommands and macros. Use the CMS command XEDIT, described in “Chapter 2” to invoke the editor prior to issuing these subcommands and macros.

An XEDIT subcommand is a command which is only valid in the environment of the editor. An XEDIT macro is a procedures language (such as REXX) program which issues XEDIT subcommands to the editor.

XEDIT subcommands and macros are generally issued from the XEDIT command line, though they may be issued from a procedures language environment such as REXX. XEDIT sets up a subcommand environment (SUBCOM) named XEDIT to process subcommands which are issued from a procedures language environment. See the XEDIT subcommands in this chapter for responses (such as return codes) that can be issued by the subcommands.

See *z/VM: REXX/VM Reference* chapter that describes “General Concepts — Issuing Subcommands from Your Program” for details on how to invoke XEDIT subcommands from a procedures language program (such as REXX). Also see the chapter that describes “System Interfaces — Calls Originating from a Clause That is an Expression” for details on how REXX passes XEDIT subcommands to the XEDIT SUBCOM.

Some XEDIT subcommands and macros may also require the presence of the EXECCOMM SUBCOM. See *z/VM: REXX/VM Reference* chapter that describes “System Interfaces — Direct Interface to Current Variables” for more about EXECCOMM SUBCOM.

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

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

CP
QUERY
SET

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

DELETE
EXPAND
HELP
LOAD
REFRESH
SORT
TYPE
XEDIT

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

- CHANGE
- PURGE
- RESET
- TRANSFER

The editor has two modes of operation: edit mode and input mode. Whenever the XEDIT command is entered, edit mode is entered; when the INPUT or REPLACE subcommands are issued with no operands, or when the POWERINP subcommand is issued, input mode is entered. For more information on how to use the editor, see *z/VM: XEDIT User's Guide*.

ADD

```
         ┌─1─┐
►►──Add──┼───┼──────────────────────────────────────────────►◄
         └─n─┘
```

Purpose

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

Operands

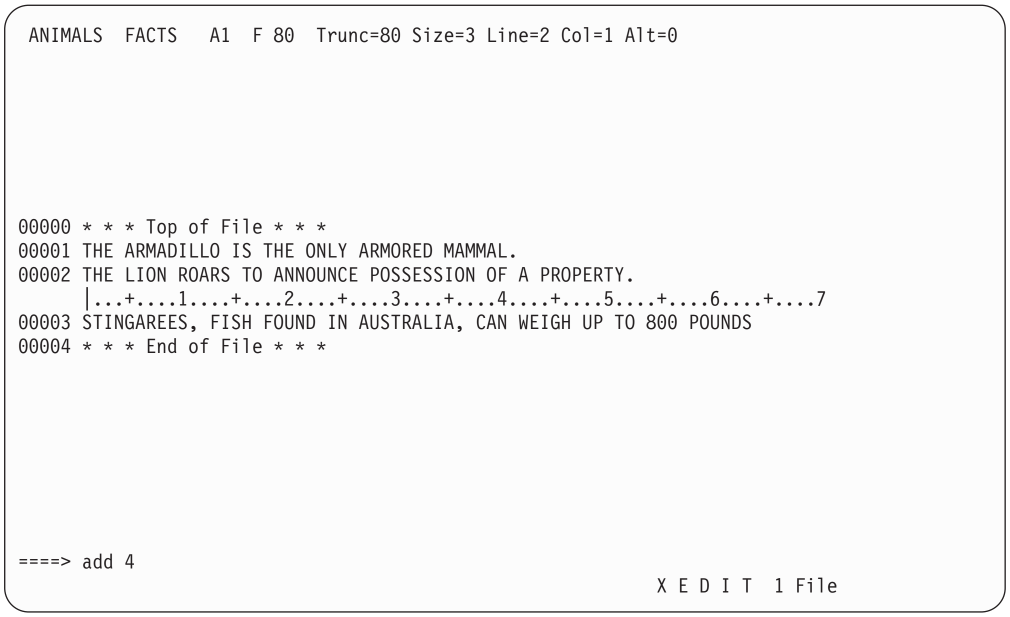
n is the number of lines you want to add. If *n* is not specified, one line is added.

Usage Notes

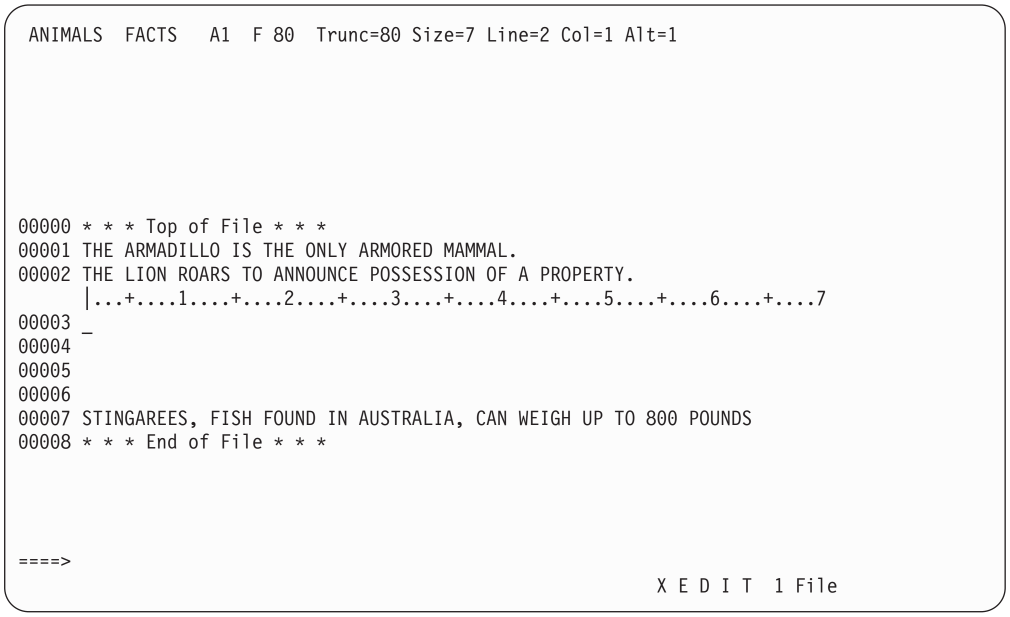
1. You can enter data in the newly-added lines at any time during the editing session. These lines remain in the file after it is saved or filed.
2. If the current line is the End of File line, the lines are added before this line.

Examples

Figure 1 on page 28 shows a before-and-after example of the ADD subcommand. In this example, four lines are being inserted after the current line (line 2).



```
 ANIMALS  FACTS   A1  F 80  Trunc=80 Size=3 Line=2 Col=1 Alt=0








00000 * * * Top of File * * *
00001 THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
00002 THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
00003 STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
00004 * * * End of File * * *









====> add 4
                                                    X E D I T  1 File
```



```
 ANIMALS  FACTS   A1  F 80  Trunc=80 Size=7 Line=2 Col=1 Alt=1








00000 * * * Top of File * * *
00001 THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
00002 THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
00003 _
00004
00005
00006
00007 STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
00008 * * * End of File * * *





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

Figure 1. ADD Subcommand — Before and After

Responses

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

By default, the prefix areas associated with the added lines are highlighted. For more information, see SET COLOR PENDING on page 253.

Each line that is added is prefilled with the current mask (see “SET MASK” on page 287).

The line pointer remains unchanged.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
543E	**Invalid number:** *number* **[RC=5]**
557S	**No more storage to insert lines [RC=4]**

where return codes are:

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

ALL (Macro)

```
           (1)
►►──ALL──┬─────────┬──────────────────────────────────────────────────────────►◄
         └─rtarget─┘
```

Notes:

1 If *rtarget* is not specified, the entire file is displayed.

Purpose

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

Operands

rtarget
: is a target that defines which lines are displayed. The target is *repeated,* that is, it is applied from the top of the file (or range) for as many times as necessary to collect all the lines in the file that correspond to the specified *rtarget*. For example, ALL/KEN/ displays all lines in the file that contain the string KEN. If *rtarget* is not specified, the entire file is displayed.

  You can specify an *rtarget* as an absolute line number, a relative displacement, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

Usage Notes

1. After you have used this macro to make changes to selected lines in a file, you can redisplay the entire file, including the changes, by issuing this macro with no *rtarget*. All lines in the file are set to a selection level of 0 and DISPLAY is set to 0 0.
2. ALL modifies the SELECT setting of all of the lines in the file and overrides the DISPLAY and SCOPE settings. ALL sets the selection level of all selected lines to 1 and all nonselected lines are set to 0. After ALL is specified, the DISPLAY setting is set to 1 1 and SCOPE is set equal to DISPLAY. If you have used the SET SCREEN subcommand, the DISPLAY setting is 1 1 for the view for which the ALL subcommand was entered, and unchanged for all other views. (See "SET SELECT" on page 333, "SET DISPLAY" on page 264, "SET SCOPE" on page 327, and "SET SCREEN" on page 329.)
3. ALL does not change the SHADOW setting. (See "SET SHADOW" on page 343) The following example shows how ALL performs when SET SHADOW is ON (the default). When SET SHADOW is OFF, a notice is not displayed to show lines are not displayed.
4. After execution of the ALL macro, the current line is as follows:

   If no target is specified or the specified target is not located, the current line remains unchanged. If the specified target is located, the first line containing the target becomes the current line.
5. Use the S (SHOW) prefix macro to redisplay the excluded lines.

6. If SET SPAN ON is in effect and the specified target spans multiple lines, only those lines containing the beginning of the target are selected for display.

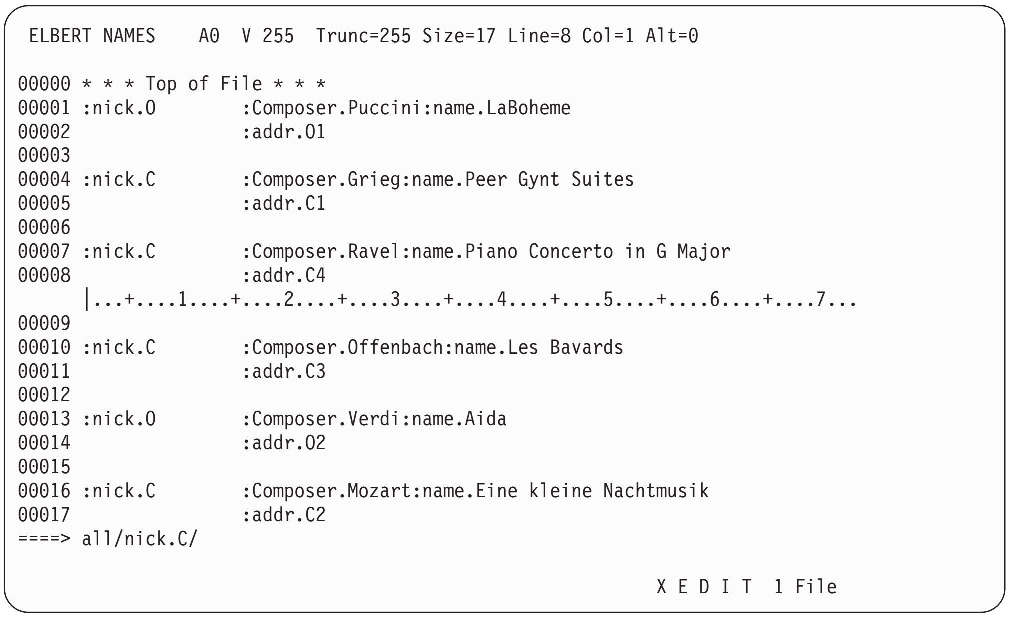
Examples

Figure 2 is an example of the ALL macro.

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

```
NICK      O for opera
          C for classical

COMPOSER  Name of Composer

NAME      Name of Composition

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

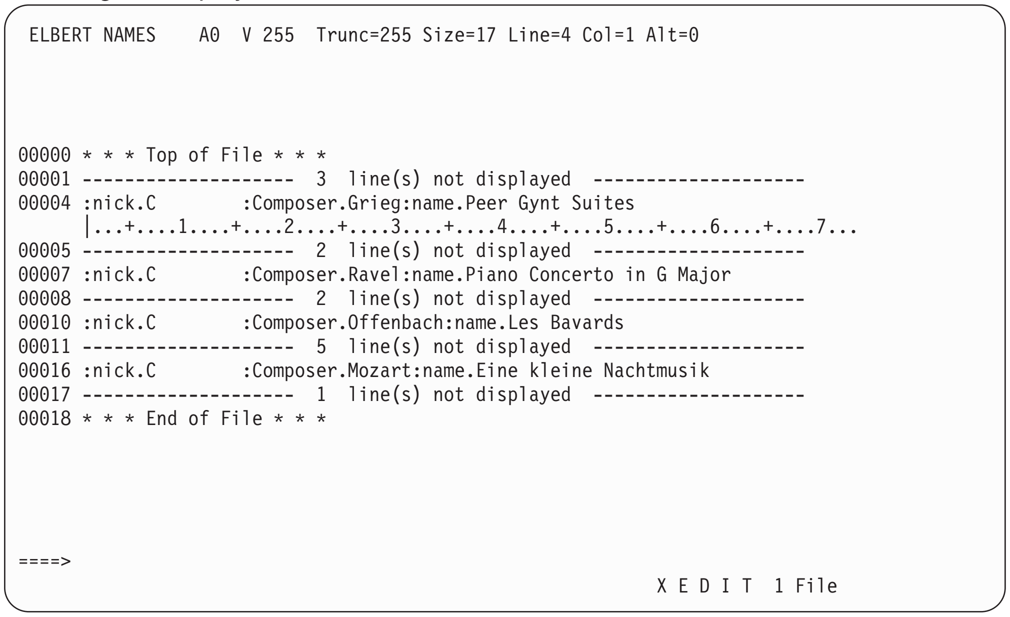
In this example, the ALL macro selects and displays all Elbert's classical records.



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

00000 * * * Top of File * * *
00001 :nick.O        :Composer.Puccini:name.LaBoheme
00002                :addr.O1
00003
00004 :nick.C        :Composer.Grieg:name.Peer Gynt Suites
00005                :addr.C1
00006
00007 :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
00008                :addr.C4
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00009
00010 :nick.C        :Composer.Offenbach:name.Les Bavards
00011                :addr.C3
00012
00013 :nick.O        :Composer.Verdi:name.Aida
00014                :addr.O2
00015
00016 :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
00017                :addr.C2
====> all/nick.C/

                                                          X E D I T  1 File
```

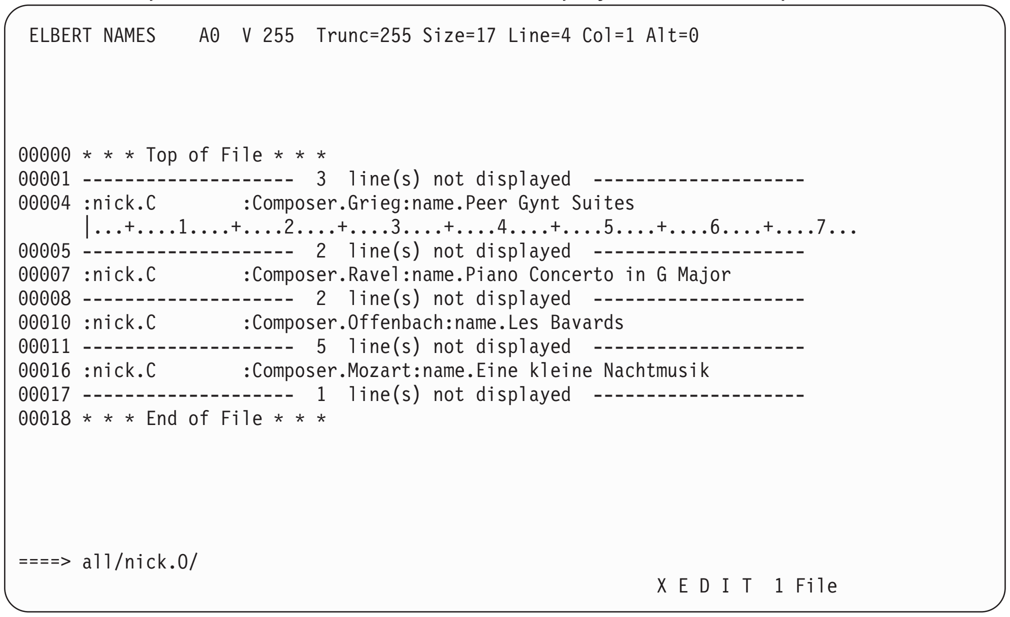
Figure 2. ALL Macro (Part 1 of 3)

Resulting file displays all classical records.



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




00000 * * * Top of File * * *
00001 --------------------  3  line(s) not displayed  --------------------
00004 :nick.C        :Composer.Grieg:name.Peer Gynt Suites
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00005 --------------------  2  line(s) not displayed  --------------------
00007 :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
00008 --------------------  2  line(s) not displayed  --------------------
00010 :nick.C        :Composer.Offenbach:name.Les Bavards
00011 --------------------  5  line(s) not displayed  --------------------
00016 :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
00017 --------------------  1  line(s) not displayed  --------------------
00018 * * * End of File * * *






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

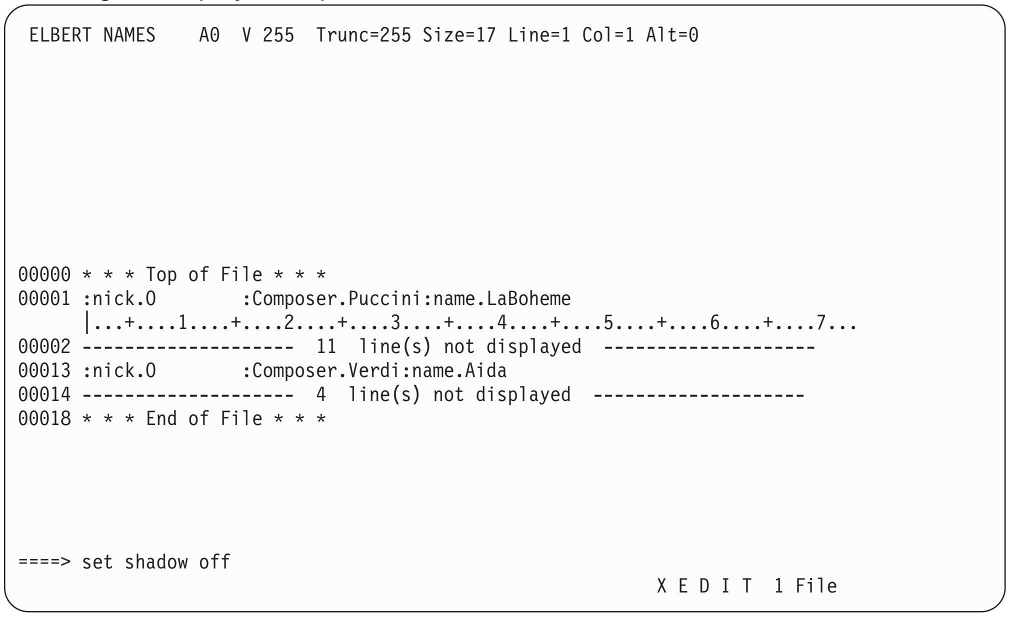
In this example, the ALL macro selects and displays all Elbert’s opera records.



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




00000 * * * Top of File * * *
00001 --------------------  3  line(s) not displayed  --------------------
00004 :nick.C        :Composer.Grieg:name.Peer Gynt Suites
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00005 --------------------  2  line(s) not displayed  --------------------
00007 :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
00008 --------------------  2  line(s) not displayed  --------------------
00010 :nick.C        :Composer.Offenbach:name.Les Bavards
00011 --------------------  5  line(s) not displayed  --------------------
00016 :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
00017 --------------------  1  line(s) not displayed  --------------------
00018 * * * End of File * * *






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

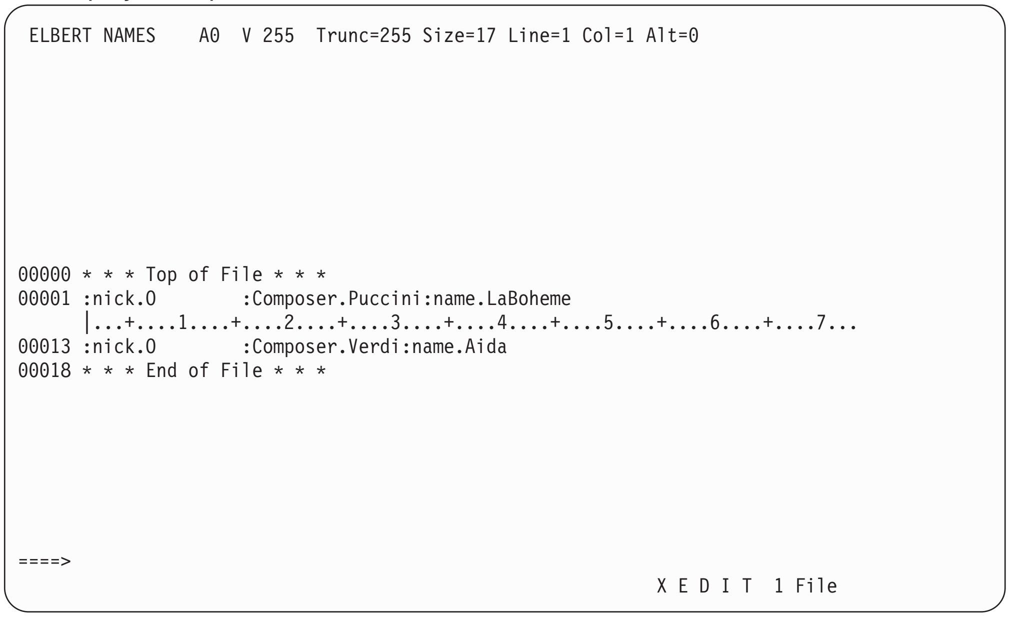
Figure 2. ALL Macro (Part 2 of 3)

Resulting file displays all opera records.



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









00000 * * * Top of File * * *
00001 :nick.O       :Composer.Puccini:name.LaBoheme
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00002 --------------------  11  line(s) not displayed  --------------------
00013 :nick.O       :Composer.Verdi:name.Aida
00014 --------------------  4  line(s) not displayed  --------------------
00018 * * * End of File * * *






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

File displays all opera records, with shadow lines set off.



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









00000 * * * Top of File * * *
00001 :nick.O       :Composer.Puccini:name.LaBoheme
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00013 :nick.O       :Composer.Verdi:name.Aida
00018 * * * End of File * * *








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

Figure 2. ALL Macro (Part 3 of 3)

Messages and Return Codes

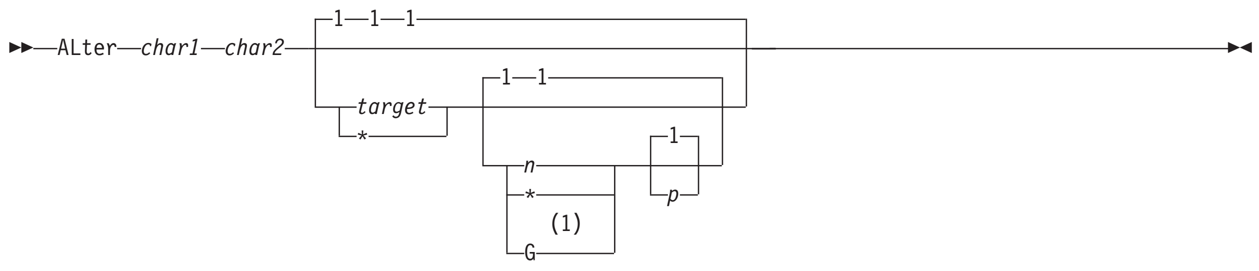
520E Invalid operand: *operand* **[RC=5]**
546E Target not found [RC=2]

where return codes are:

0	Normal
2	Target not found

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

ALTER (Macro)



Notes:

1 You can only specify G, if you specified *target* as a number.

Purpose

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.

Operands

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

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

target
: defines the number of lines to be searched for *char1*. The search for *char1* starts with the current line and continues 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” on page 317.) If *target* is not specified, only the current line is altered.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see “LOCATE” on page 147 and *z/VM: XEDIT 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 *n* is not specified, only one occurrence of *char1* in each line is altered. For compatibility with the CMS editor (EDIT), you can specify the G (Global) operand, but only when you specify *target* as a number.

p
: specifies the relative number of the first occurrence of *char1* to be altered in each line examined. If *p* is not specified, the alteration starts with the first occurrence of *char1* in a line.

Examples

This section shows an example of the ALTER macro. For more information, see *z/VM: XEDIT User’s Guide*.

To produce compound characters on printed output, you can use ALTER to change a special character to a backspace character.

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 looks like this:

```
Please underline THIS
```

Responses

The column pointer remains unchanged.

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

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

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

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

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

Messages and Return Codes

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

where return codes are:

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

BACKWARD

```
         ┌─1─┐
►►──BAckward──┼───┼──────────────────────────────────────────────────────────►◄
         ├─n─┤
         └─*─┘
```

Purpose

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

Operands

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 *n* is not specified, the screen scrolls back one display.

Usage Notes

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

Examples

In the following example, the BACKWARD subcommand is being used to scroll backward 3 screens.

```
===== I want to scroll back 3 screens.

ba 3
```

Your display screen will scroll back 3 screens.

Messages and Return Codes

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

where return codes are:

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

BOTTOM

```
►►──Bottom──────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the BOTTOM subcommand to move the line pointer to the line above the null End of File or End of Range (see "SET RANGE" on page 317) line.

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).
3. If you issue the BOTTOM subcommand while editing a file containing no records, the current line remains the same.

Examples

In the following example, entering the BOTTOM subcommand makes the last line of the file the new current line.

Current Line:

```
00014 My file has 23 lines.  I want line 23 to be my current line.
bottom
```

Current Line:

```
00023 The last line of my file is now my current line.
```

The current line is 23 (last line in my file).

Responses

If you issue the BOTTOM subcommand while editing a file containing no records, the following message is displayed:

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

Messages and Return Codes

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

where return codes are:

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

CANCEL (Macro)

```
►►──CANCEL──────────────────────────────────────────────────────────►◄
```

Purpose

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

Usage Notes

1. The QUIT that is issued against all the files is either *protected* or *unprotected*, depending on the defined synonyms (see "QUIT" on page 202). If the QUIT subcommand has been defined to perform a protected QUIT, the CANCEL macro quits all unmodified files but issues a warning message for each modified file, leaving the user in edit mode. If all the files being canceled were unmodified, the CANCEL macro causes an immediate exit from the editor.

Responses

(if protected QUIT is defined):

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

Messages and Return Codes

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

where return codes are:

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

CAPPEND (Macro)

```
                (1)
►►──CAppend──┬──────┬──────────────────────────────────────────────────────►◄
             └─text─┘
```

Notes:

1 If *text* is not specified, the column pointer is placed under the first trailing blank.

Purpose

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

Operands

text
: is the text to be appended to the end of the current line. Whether or not *text* is specified, the column pointer is placed under the first trailing blank.

Usage Notes

1. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.
2. The text operand starts with the first character following the blank delimiter after the subcommand name. The maximum length of *text* is 240 bytes.
3. Column pointer movement is affected by the current zone setting. See "SET ZONE" on page 371 for more complete information.

Examples

This section shows examples of the CAPPEND macro. For more information, see *z/VM: XEDIT User's Guide*.

Example 1

In this example, one blank is inserted between the subcommand name and operand.

Current Line:

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

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

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

Example 2

In this example, two blanks are inserted between the subcommand name and operand.

Current Line:

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

cappend  and he stoppeth one of three.

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

Responses

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

Messages and Return Codes

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

where return codes are:

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

CDELETE

```
           ┌─1─────────────┐
►►──CDelete─┼───────────────┼──────────────────────────────────────────────►◄
           └─column_target─┘
```

Purpose

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

Operands

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. If *column_target* is not specified, one character is deleted.

For a complete description of column-targets, see "CLOCATE" on page 53.

Usage Notes

1. Use the CLOCATE subcommand to move the column pointer to the column at which you want deletion to begin.
2. As with all column-targets, the following SET options have an effect on the column-target search:
   SET ARBCHAR
   SET CASE
   SET ETARBCH
   SET SPAN
   SET VARBLANK
3. When SET STREAM OFF is in effect, only the current line is searched for the string to be deleted. When SET STREAM ON is in effect, the editor deletes starting at the character following the column pointer and continues to the end of the file (or range), or to the top of the file (or range) if the search is in a backward direction, until the target is reached. In this case, several lines may be deleted.

Examples

This section shows examples of the CDELETE subcommand. For more information, see *z/VM: XEDIT User's Guide*.

Example 1

In this example, starting at the column pointer, the characters on the current line are deleted using CDELETE.

Current Line:

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

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

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

Example 2

In this example, characters are deleted from the current column and preceding columns by using CDELETE.

Current Line:

```
===== A dialect is considered a language if it is used in newspapers.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
```

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

```
===== A dialect is considered a language if used in newspapers.
      <...+....1....+....2....+....3....+....4..|.+....5....+....6....+....7...
```

Example 3

In this example, characters are deleted from the column pointer to the first character of the string using CDELETE.

Current Line:

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

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

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

Example 4

In this example, characters are deleted from the column pointer up to the first character of the string using CDELETE.

Current Line:

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

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

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

Responses

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

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

Messages and Return Codes

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

where return codes are:

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

CFIRST

```
►►──CFirst─────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the CFIRST subcommand to move the column pointer to the beginning of the zone (see “SET ZONE” on page 371).

Usage Notes

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

Examples

In the following example, the column pointer will be moved from column 41 to the beginning of the zone using CFIRST.

Current Line:

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

cfirst

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

Now the column pointer is reset to column 1.

For more information, see *z/VM: XEDIT User’s Guide*.

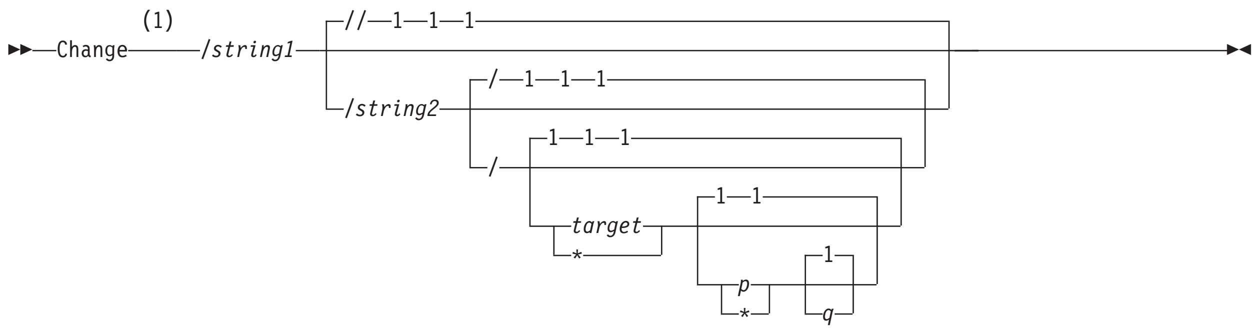
Messages and Return Codes

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

where return codes are:

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

CHANGE



Notes:

1 If *string1* or *string2* is omitted, it is assumed to be a null string.

Purpose

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.

Operands

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

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

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

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

You can specify target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT 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 .z/VM.CMS.` changes `z/VM` to `CMS`

   To change blanks to nulls when SET HEX ON is in effect:

   ```
   change ?x'40'?x'00'?* *
   ```
2. If *string2* is longer than *string1* and if SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. If a line is spilled, no additional changes are made on that line.

   If *string2* is shorter than *string1*, characters are shifted left (from the truncation column), and the line is padded with blanks (up to the truncation column).
3. If *string1* is represented as a null string, *string2* is inserted in the line, starting at the beginning of the zone, which may or may not be column 1.
4. Using CHANGE with SET ARBCHAR ON:

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

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

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

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

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

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

   For additional examples of using CHANGE with SET ARBCHAR, see page 48 and “SET ARBCHAR” on page 239.

   To use CHANGE with SET ETARBCH see Appendix F, “Using Double-Byte Character Sets,” on page 467.
5. Using SET CASE and CHANGE:

   String1 should be typed exactly as it appears in the file to be changed (even with SET CASE MIXED IGNORE).
6. Using SET STAY and CHANGE:

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

   a. Is unchanged, if SET STAY ON is in effect
   b. Moves to the last line scanned, if SET STAY OFF is in effect (the default)
7. Using SET ZONE and CHANGE:

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

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

This replaces all characters from zone 1 through zone 2 (not the truncation column) with characters xy. Characters from zone 2 to the truncation column are shifted left and the line is padded with blanks (up to the truncation column).

8. Using CHANGE with SET SPAN ON:

   The search for *string1* is not affected by any of the settings you have using the SET SPAN subcommand to span several lines (the search is executed as if SET SPAN OFF were in effect).
9. Using CHANGE with SET VARBLANK ON:

   The search for *string1* is not affected by any of the settings you have using SET VARBLANK to control the number of blank characters between two words in a target search (the search is executed as if SET VARBLANK OFF were in effect).
10. The CHANGE subcommand updates the LASTLORC buffer. See “SET LASTLORC” on page 281.

Examples

This section shows examples of the CHANGE subcommand. For more information, see *z/VM: XEDIT User’s Guide*.

Example 1

In this example, CHANGE is used to change the first occurrence in the current line.

Current Line:

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

change/rose/daisy/

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

Example 2

In this example, CHANGE is used to change all occurrences in the current line.

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

change/rose/daisy/ 1 *

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

Example 3

In this example, CHANGE is used to change every occurrence of “daisy” to “rose” in every line of the file, beginning with the current line.

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

Example 4

In this example, CHANGE is used to insert characters in column 1.

Current Line:

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

```
change//Mr. /
```

(insert ″Mr. ″ in column 1)

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

Example 5

In these examples, CHANGE is being used with SET ARBCHAR ON.

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

```
change/'$'/"$"/
```

(change single quotation marks to double 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."
```

Responses

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

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

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

Messages and Return Codes

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

where return codes are:

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

CINSERT

```
►►──CInsert──text──────────────────────────────────────────────────────────►◄
```

Purpose

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

Operands

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

Usage Notes

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

Examples

In the following example, the column pointer is moved and text is inserted starting at the column pointer. For more information, see *z/VM: XEDIT User's Guide*.

Current Line:

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

```
cl /high/
```

(move the column pointer)

```
ci exactly 29,000 feet
```

(one blank is entered after "feet" for spacing)

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

Responses

The column pointer remains unchanged.

Messages and Return Codes

503E {Truncated|Spilled} [RC=3]

545E Missing operand(s) [RC=5]
585E No line(s) changed [RC=4]

where return codes are:

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

CLAST

```
►►──CLAst────────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the CLAST subcommand to move the column pointer to the end of the zone (see "SET ZONE" on page 371).

Examples

In the following example, the column pointer is moved to the end of the zone previously specified by SET ZONE.

```
set zone 1 20
```

Current Line:

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


clast

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

Messages and Return Codes

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

where return codes are:

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

CLOCATE

```
►►──CLocate──column_target──────────────────────────────────────────────────►◄
```

Purpose

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 successively finds *all* occurrences of a character string and moves the column pointer if the string is found.

Operands

column_target
can be specified as an absolute column number, a relative column number, a string expression, or a complex string expression.

A *column_target* is a specialized operand used only in the CLOCATE and CDELETE subcommands.

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

Therefore, 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 moves the column pointer to a specified column or locates a specified string; the column pointer is moved if the string is found.

You can specify a *column_target* in the following ways:

1. An *absolute column number* 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 a subcommand that starts its operation at the column pointer, for example, JOIN COLUMN.
2. A *relative column number* is expressed as an integer that can be preceded by a plus (+) or minus (–) sign, which indicates moving the column pointer right (+) or left (–). If the sign is omitted, a plus (+) is assumed.

For example:

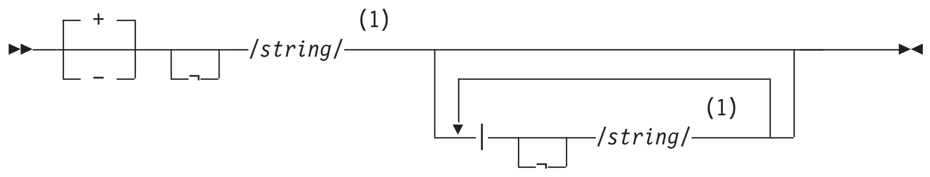
```
clocate +2
clocate  2
```

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

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

   If SET HEX ON is in effect, you can specify a string in hexadecimal notation, and the EBCDIC equivalent is searched for.

   The format of a string expression is:



Notes:

1 The final delimiter (/) is optional after the last *string*. Leading or trailing blanks are considered part of *string*.

+ or –
Right (+) or left (–) search (right is the default).

¬ *NOT* symbol. (Locate something that is not the specified *string*.)

string
Character (or hexadecimal) string. The trailing delimiter may be necessary in certain circumstances. For example, if the first *string* has trailing blanks, use the trailing delimiter to indicate where the string ends.

| *OR* symbol (vertical bar). (Locate any of the *string*s, separated by OR symbols, starting with the first *string* specified.)

For example:

`clocate /horse/`
searches the file for the first occurrence of *horse*, starting at the first character after the column pointer.

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

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

`clocate /horse/|/buggy/`

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

```
clocate //
```

advances the column pointer one column.

4. A *complex string expression* has the same format as a string expression. A complex string is a string associated with one or more of the following SET subcommand options:

   ARBCHAR

   CASE

   ETARBCH

   SPAN

   VARBLANK

   See “LOCATE” on page 147 for a description of these options.

Usage Notes

1. SET STREAM is meaningful only with *column_targets*

   ```
   stream on
   ```

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

   If SET WRAP is also ON (OFF is the default) and the editor *wraps around* the file (see “SET WRAP” on page 370) and reaches the top of file (or range) or end of file (or range) during a CLOCATE, the following warning message is displayed:

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

   ```
   stream off
   ```

   specifies the search is limited to the current line (or zones within the line).
2. The CLOCATE subcommand updates the LASTLORC buffer. See “SET LASTLORC” on page 281.
3. For more information on the CLOCATE subcommand, see Appendix B, “Effects of Selective Line Editing Subcommands,” on page 455.

Examples

This section shows examples of the CLOCATE subcommand. For more information, see *z/VM: XEDIT User’s Guide*.

Example 1

In the following example, CLOCATE is used to move the column target using an absolute column number.

Current Line:

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

clocate  :6

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

Example 2

In the following example, CLOCATE is used to move the column target using a relative column number.

Current Line:

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

clocate  +6

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

Example 3

In the following example, CLOCATE is used to move the column target using a string expression.

Current Line:

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

clocate /customs/

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

Example 4

In the following example, CLOCATE is used to move the column target using a complex string expression.

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....+..
```

Messages and Return Codes

520E Invalid operand: ***operand*** **[RC=5]**
545E Missing operand(s) [RC=5]
546E Target not found [RC=2]
592W Wrapped ....

700E Logical AND operator & not valid for *column_targets* **[RC=5]**

where return codes are:

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

CMS

```
             (1)
>>──CMS──┬─────────┬────────────────────────────────────────────────────────────><
         └─command─┘
```

Notes:

1 If *command* is not specified, the editor enters CMS subset mode.

Purpose

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

Operands

command
: is any CMS command. If *command* is not specified, the editor enters CMS subset mode.

Usage Notes

1. If you enter the CMS subcommand *without* an operand, you enter CMS subset mode. For a description of CMS subset mode, see *z/VM: CMS User's Guide*. To return to edit mode, use the CMS subset command RETURN.
2. If you try to execute a CMS command that terminates abnormally, changes made during your editing session might be lost. Save the input you have entered before you use the CMS subcommand.
3. Preface any CMS command with *CMS* to prevent XEDIT from mistaking the CMS command for an XEDIT subcommand.
4. Some CMS commands issue either informational messages or error messages to the XEDIT environment. While in CMS subset mode, you do not see these messages until you return to XEDIT unless MSGLINE is set OFF or the maximum number of message lines has been exceeded. However, you see the return code associated with the message as part of the CMS ready message.

Examples

In the following example, the editor will send the ERASE command to CMS for execution. CMS will erase a file called FILE SCRIPT.

```
cms erase file script
```

Responses

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

```
CMS subset
```

This indicates you are in CMS subset mode. If full-screen CMS is ON (through the SET FULLSCREEN ON command), the message appears in the CMS window connected to the virtual screen. Also, the STATUS window is popped. If full-screen

CMS is OFF, the editor clears the screen before issuing the CMS SUBSET message. To return to the XEDIT environment when full-screen CMS is ON **or** OFF, use the CMS RETURN command.

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

If you are in a GUI environment using XEDIT, the CMS subcommand will fail without an operand. Use your terminal emulator if a CMS input screen is needed.

If full-screen CMS is ON and the CMS command displays text, the text appears in the CMSOUT window. (Text appears in the CMSOUT window by default; you can route the text to another window with the CMS VSCREEN ROUTE command, explained in *z/VM: CMS Commands and Utilities Reference*). To see all the text in the virtual screen, you can use the CMS WINDOW FORWARD or WINDOW BACKWARD command. The screen is cleared automatically when you scroll to the bottom of the virtual screen. Alternately, you can clear the screen with the CMS WINDOW DROP command. If you delete the CMSOUT window, you do not see messages passed to CMS.

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

If you are in a GUI environment, CMS responses display in the CMS OUTPUT window.

Messages and Return Codes

512E	**Invalid subset command [RC=-2]**
513E	**Unknown CP/CMS command [RC=-3]**
514E	**Return code** *nn* **from** *command* **[RC=±*nn*]**
642E	**Subcommand or option is not valid in GUI mode [RC=5]**

where return codes are:

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

CMSG

```
           (1)
►►──CMSG──┬──────┬──────────────────────────────────────►◄
          └─text─┘
```

Notes:

1 If *text* is not specified, the command line is reset to a blank line.

Purpose

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.

Operands

text
is the text of the message to be displayed. If *text* is not specified, the command line is reset to a blank line.

Usage Notes

The CMSG subcommand has no effect when it is issued:

1. To a typewriter terminal
2. When the command line is off
3. When the stack is not empty

Notes for Macro Writers

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

Responses

The message is displayed in the command line.

Messages and Return Codes

This subcommand issues no messages, but it issues return codes as follows:

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

COMMAND

```
►►──COMMAND──┬────────────┬──────────────────────────────────────────────────►◄
             └─subcommand─┘
```

Purpose

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

Operands

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

Usage Notes

1. The editor executes the specified subcommand even if SET SYNONYM ON or SET MACRO ON is in effect. 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 the editor does not recognize are transmitted to CMS or to CP.

Examples

In the following example, the PRESERVE subcommand is executed, even if a synonym or macro with the same name exists.

```
command pres
```

Responses

Any response from the executed subcommand is displayed.

Messages and Return Codes

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

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

where return codes are:

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

COMPRESS

```
           ┌─1──────┐
►►──COMPress─┼────────┼──────────────────────────────────────────────────────────────►◄
           ├─target─┤
           └─*──────┘
```

Purpose

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

Operands

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

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

Usage Notes

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

   Using this sequence of commands:

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

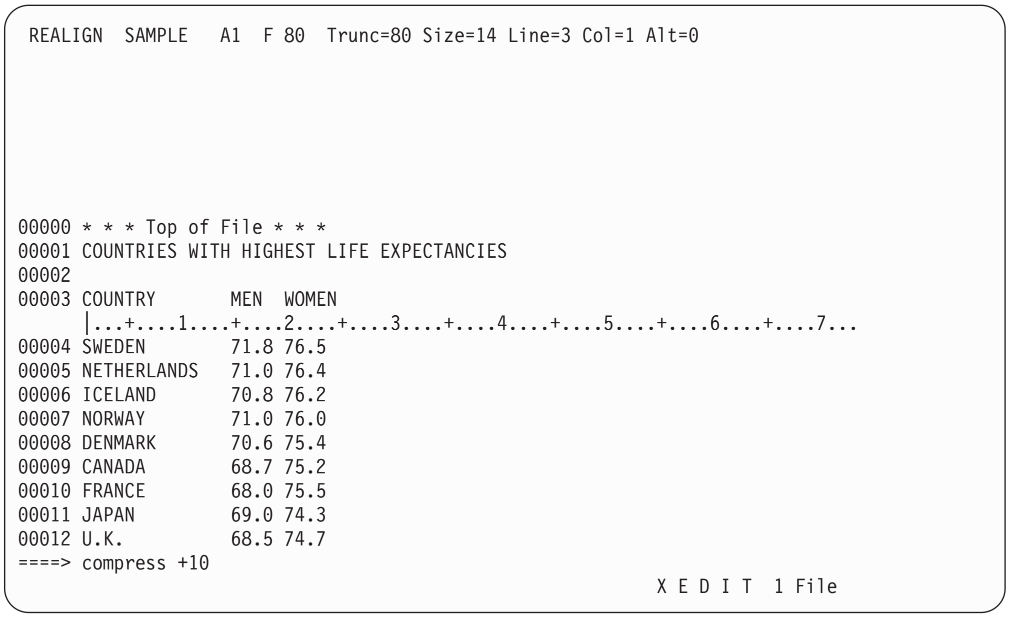
   you can realign an entire table.
2. For COMPRESS to work properly, lines containing backspace characters (X'16') must have been entered into the file with SET IMAGE CANON, that is, compound characters ordered with backspaces arranged singly between the characters that overlay each other.
3. Lines are compressed according to the current SET TABS setting. The COMPRESS subcommand removes all contiguous blank characters (or whatever character is defined as a filler character by the SET FILLER subcommand) that immediately precede the current tab columns. It replaces each group of blank (or filler) characters with a tab character (X'05').

Examples

This section shows examples of the COMPRESS and EXPAND subcommands to realign data in a table.

Example 1

Figure 3 is an example of using COMPRESS to prepare the table (which has the current tab settings of 1 15 20) for automatic repositioning with new tab column settings. The current line is 3.

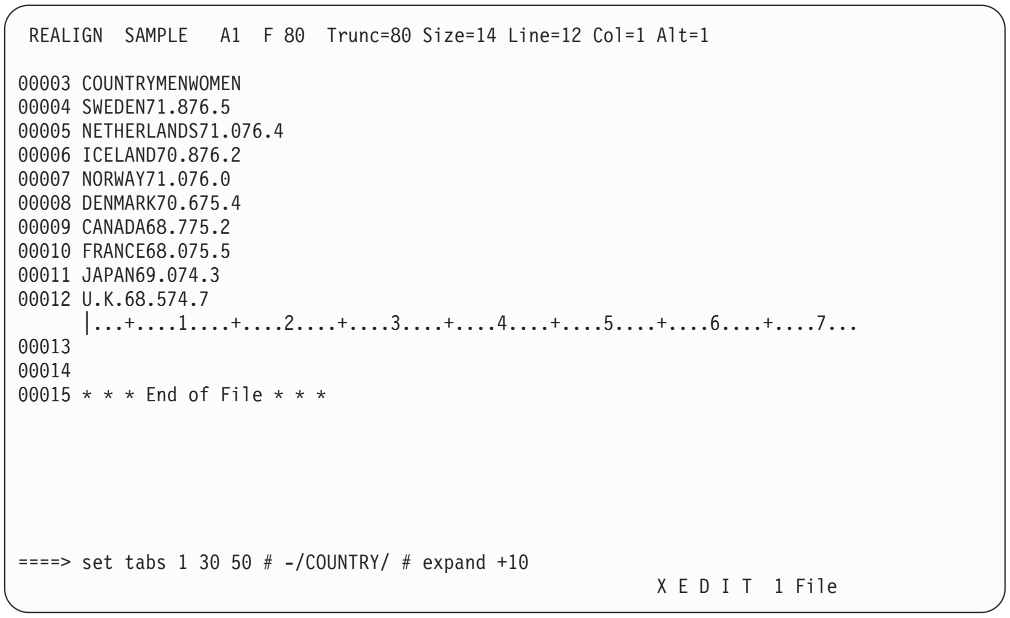


```
 REALIGN  SAMPLE   A1  F 80  Trunc=80 Size=14 Line=3 Col=1 Alt=0








00000 * * * Top of File * * *
00001 COUNTRIES WITH HIGHEST LIFE EXPECTANCIES
00002
00003 COUNTRY       MEN  WOMEN
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00004 SWEDEN        71.8 76.5
00005 NETHERLANDS   71.0 76.4
00006 ICELAND       70.8 76.2
00007 NORWAY        71.0 76.0
00008 DENMARK       70.6 75.4
00009 CANADA        68.7 75.2
00010 FRANCE        68.0 75.5
00011 JAPAN         69.0 74.3
00012 U.K.          68.5 74.7
====> compress +10
                                                            X E D I T  1 File
```

Figure 3. Using COMPRESS to Realign a Table

Example 2

Figure 4 is an example of setting up new tabs and using EXPAND to reposition the data in the new tab columns.

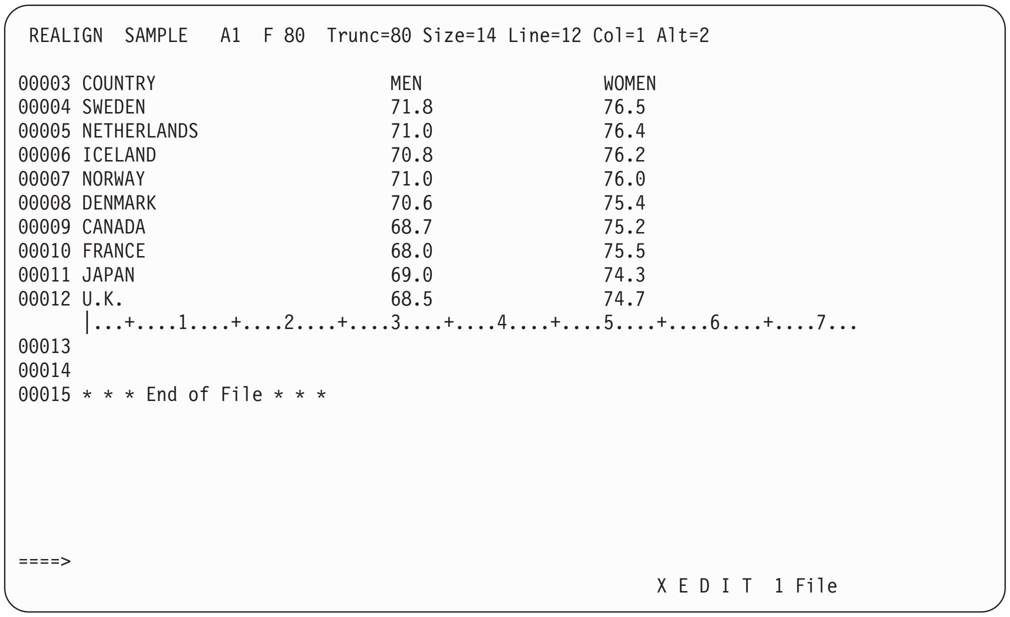


```
 REALIGN  SAMPLE   A1  F 80  Trunc=80 Size=14 Line=12 Col=1 Alt=1

00003 COUNTRYMENWOMEN
00004 SWEDEN71.876.5
00005 NETHERLANDS71.076.4
00006 ICELAND70.876.2
00007 NORWAY71.076.0
00008 DENMARK70.675.4
00009 CANADA68.775.2
00010 FRANCE68.075.5
00011 JAPAN69.074.3
00012 U.K.68.574.7
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00013
00014
00015 * * * End of File * * *






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

Figure 4. Setting up New Tabs and Using EXPAND

Example 3

Figure 5 on page 64 is an example of the realigned table.



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

00003 COUNTRY                     MEN                WOMEN
00004 SWEDEN                      71.8               76.5
00005 NETHERLANDS                 71.0               76.4
00006 ICELAND                     70.8               76.2
00007 NORWAY                      71.0               76.0
00008 DENMARK                     70.6               75.4
00009 CANADA                      68.7               75.2
00010 FRANCE                      68.0               75.5
00011 JAPAN                       69.0               74.3
00012 U.K.                        68.5               74.7
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00013
00014
00015 * * * End of File * * *






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

Figure 5. Realigned Table

Responses

1. If you specify a compress on multiple lines and the compress occurs, the current line pointer:
   a. Is unchanged, if SET STAY ON is in effect
   b. Moves to the last line compressed, if SET STAY OFF is in effect (the default)
2. The data in a compressed line shifts left, reflecting the removal of blanks.

Messages and Return Codes

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

where return codes are:

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

COPY

```
►►──COpy──target1──target2──────────────────────────────────────────────────►◄
```

Purpose

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

Operands

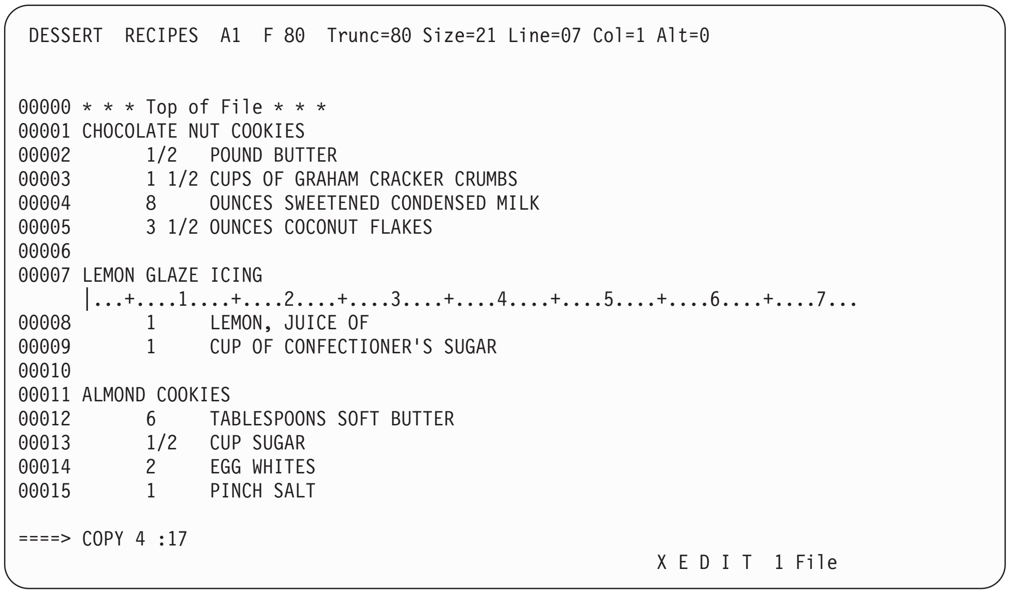
target1
defines the number of lines to be copied. Lines are copied starting with the current line, up to but not including *target1*.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

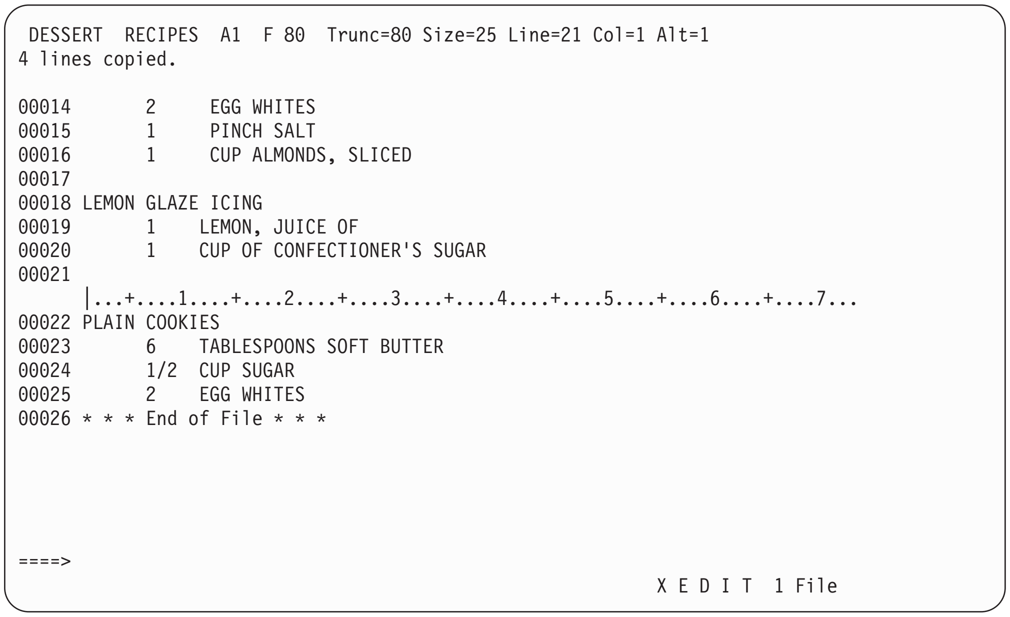
target2
defines the line after which the data is to be copied.

Examples

Figure 6 on page 66 is a before-and-after example of the COPY subcommand.



```
 DESSERT  RECIPES  A1  F 80  Trunc=80 Size=21 Line=07 Col=1 Alt=0


00000 * * * Top of File * * *
00001 CHOCOLATE NUT COOKIES
00002      1/2   POUND BUTTER
00003      1 1/2 CUPS OF GRAHAM CRACKER CRUMBS
00004      8     OUNCES SWEETENED CONDENSED MILK
00005      3 1/2 OUNCES COCONUT FLAKES
00006
00007 LEMON GLAZE ICING
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00008      1     LEMON, JUICE OF
00009      1     CUP OF CONFECTIONER'S SUGAR
00010
00011 ALMOND COOKIES
00012      6     TABLESPOONS SOFT BUTTER
00013      1/2   CUP SUGAR
00014      2     EGG WHITES
00015      1     PINCH SALT

====> COPY 4 :17
                                                            X E D I T  1 File
```



```
 DESSERT  RECIPES  A1  F 80  Trunc=80 Size=25 Line=21 Col=1 Alt=1
4 lines copied.

00014      2     EGG WHITES
00015      1     PINCH SALT
00016      1     CUP ALMONDS, SLICED
00017
00018 LEMON GLAZE ICING
00019      1    LEMON, JUICE OF
00020      1    CUP OF CONFECTIONER'S SUGAR
00021
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00022 PLAIN COOKIES
00023      6    TABLESPOONS SOFT BUTTER
00024      1/2  CUP SUGAR
00025      2    EGG WHITES
00026 * * * End of File * * *





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

Figure 6. COPY Subcommand — Before and After

Responses

The last line copied becomes the new current line.

The editor displays the following message:

`506I  nn lines copied`

Messages and Return Codes

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

where return codes are:

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

COUNT

```
                    ┌─1──────┐
►►──COUnt──/string/─┼────────┼──────────────────────────────────────────────►◄
                    ├─target─┤
                    └─*──────┘
```

Purpose

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.

Operands

string
is the character string to be counted.

target
defines the number of lines to be searched. The search begins with the current line and continues up to, but does not include, the target line. If *target* is not specified, only the current line is searched.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

***** the search continues to the end of file (or the end of range—see "SET RANGE" on page 317).

Usage Notes

1. The count corresponds to the number of strings that would be changed by a CHANGE subcommand.
2. Arbitrary characters (see "SET ARBCHAR" on page 239) can be specified in the /*string*/ operand.

   For example:

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

   counts all of the expressions enclosed in parentheses.
3. In a macro, you can use EXTRACT/LASTMSG/ to get the number of occurrences.
4. The CP EMSG and XEDIT MSGMODE settings are handled differently for the message COUNT issues containing the number of occurrences of a string.

   The LASTMSG buffer is always updated with this message regardless of the EMSG and MSGMODE settings (even if EMSG OFF or MSGMODE ON (or OFF) SHORT is in effect). When the message is displayed or the LASTMSG buffer is updated, the message text is always included in the message (even if EMSG CODE is in effect).
5. The *string* should be typed exactly as it appears in the file to be counted regardless of the CASE setting.

Examples

In the following example, the COUNT subcommand is used to display the number of times a specific character string appears. For more information, see *z/VM: XEDIT User's Guide*.

Current Line:

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

```
count/rose/
```

Response:

```
522I  3 occurrences
```

Responses

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

1. Is unchanged, if SET STAY ON is in effect
2. Moves to the last line scanned, if SET STAY OFF is in effect (the default)

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

```
522I  nn occurrences
```

Messages and Return Codes

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

where return codes are:

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

COVERLAY

```
►►──COVerlay──text──────────────────────────────────────────────────────────►◄
```

Purpose

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.

Operands

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 do *not* overlay corresponding characters in the current line.

   For example:

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

2. An underscore character (_) in the text operand places a blank in the corresponding character position in the current line.

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

   Use the COVERLAY command carefully if a line contains underscored words or other compound characters.
3. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.
4. If the column pointer is at Zone1–1 (TOL) or Zone2+1 (EOL), no characters are overlaid.

Examples

In the following example, COVERLAY selectively replaces characters in the current line with the corresponding characters that are keyed in. Since the replacement starts at the column pointer, the CLOCATE subcommand is used to move the pointer to the correct location.

Current Line:

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

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

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

Messages and Return Codes

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

where return codes are:

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

CP

```
        (1)
►►──CP──┬─────────┬──────────────────────────────────────────────►◄
        └─command─┘
```

Notes:

1 If *command* is not specified, the editor enters the CP console function mode.

Purpose

Use the CP subcommand to transmit commands to the Control Program (CP) environment during an editing session.

Operands

command
: 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 *command* is not specified, the editor enters the CP console function mode, where you can enter CP commands without preceding each command with CP. To return to the edit environment, enter the CP command BEGIN.

Usage Notes

1. Preface any CP command with *CP* to prevent the editor from mistaking the CP command for an XEDIT subcommand.

Responses

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

If full-screen CMS is ON and the CP command displays text, the text appears in the CMSOUT window. (Text appears in the CMSOUT window by default; you can route the text to another window with the CMS VSCREEN ROUTE command, explained in *z/VM: CMS Commands and Utilities Reference*). To see all of the text in the virtual screen, you can use the CMS WINDOW FORWARD or WINDOW BACKWARD command. The screen is cleared automatically when you scroll to the bottom of the virtual screen. Alternately, you can clear the screen with the CMS WINDOW DROP command. If you delete the CMSOUT window, you do not see messages that are passed to CMS.

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

Issuing the CP subcommand without an operand will fail if you are in a GUI environment. Use your terminal emulator for CP console function mode.

Messages and Return Codes

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

642E Subcommand or option is not valid in GUI mode [RC=5]

where return codes are:

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

CREPLACE

```
►►──CReplace──text──────────────────────────────────────────────────────────►◄
```

Purpose

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.

Operands

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

Usage Notes

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

Examples

In the following example, CREPLACE replaces characters in the current line with specified characters, starting at the column pointer. The CLOCATE subcommand is used first to move the column pointer.

Current Line:

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

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

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

Messages and Return Codes

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

where return codes are:

0	Normal

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

CURSOR

```
                   ┌─1─────┐
►►──CURsor──┬─CMdline─┼───────┼─────────┬─┬─────────────┬──────────────────────────►◄
            │         └─colno─┘         │ └─Priority──n─┘
            ├─Column────────────────────┤
            ├─Home──────────────────────┤
            │                   (1)     │
            │                  ┌───────┐│
            └──┬─File───┬─lineno─┼───────┼┘
               └─Screen─┘        └─colno─┘
```

Notes:

1 For the default information, refer to the *colno* description.

Purpose

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

Operands

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

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

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

File
moves the cursor to a line number 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).
- FILE, *colno* places the cursor relative to the beginning of a file line. 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.

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

Priority *n*

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

See page 422 for a complete list of the priorities for all prefix subcommands and macros, screen changes, and the ENTER key.

Initial Setting

The initial placement of the cursor is based on the initial setting for SET ENTER:

```
CURSOR CMDLINE 1 PRIORITY 30
```

Usage Notes

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

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

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

Notes for Macro Writers

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

   ```
   cursor file 700
   ```

   file line 700 must appear on the current logical screen.

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

   When you use CURSOR COLUMN, the column position must appear on the current logical screen. For example, if using SET VERIFY, you issue:

   ```
   clocate 10
   set verify 20 30
   cursor column
   ```

   an error message is displayed because column 10 no longer appears on your current logical screen. Other XEDIT subcommands which can affect the contents of the logical screen include RIGHT and LEFT.
2. A macro can indicate where the cursor should be positioned as well as the priority for this cursor movement. As a result, when multiple prefix subcommands are issued and a macro is executed on the command line, the cursor is positioned at the location specified with the highest priority.

   Note: The cursor position is updated ONLY when a higher cursor priority is met. Therefore, if two prefix subcommands are specified that have the same cursor priority, the first one determines the cursor placement. Likewise, if a prefix subcommand is issued and a macro is executed on the command line, both with equal cursor priorities, the prefix subcommand determines the cursor placement.
3. If multiple CURSOR subcommands are issued, the priority of each is checked and the cursor position is updated to reflect the new setting, which corresponds to the command with the highest priority. The cursor remains in the screen it was in when ENTER is pressed; therefore, a CURSOR subcommand in another logical screen has no effect on cursor placement.

Messages and Return Codes

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

where return codes are:

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

DELETE

```
          ┌─1──────┐
►►──DELete──┼────────┼──────────────────────────────────────────────────────►◄
          ├─target─┤
          └─*──────┘
```

Purpose

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

Operands

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 *target* is not specified, only the current line is deleted.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

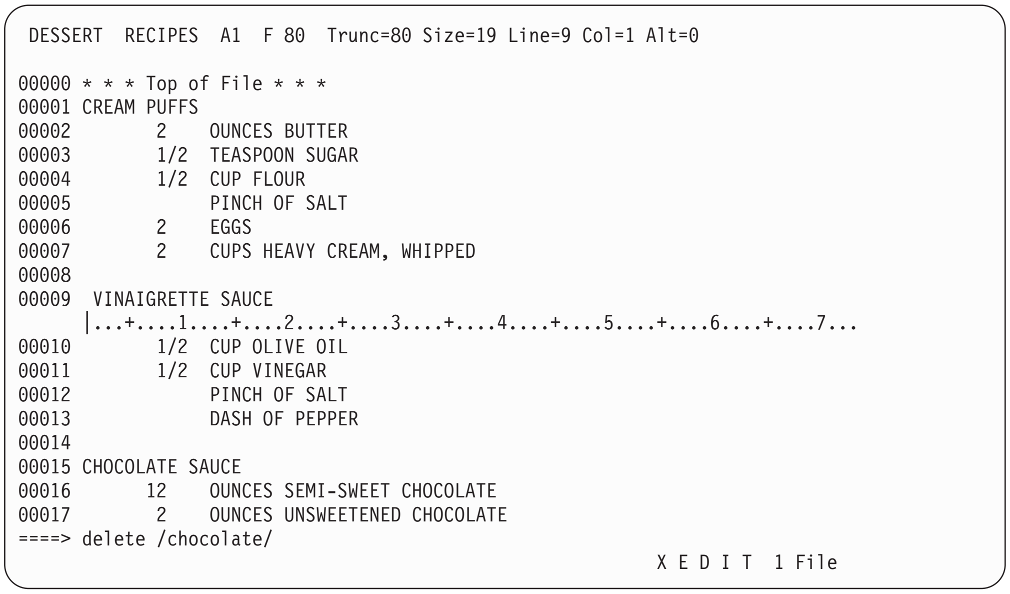
* deletes the rest of the file.

Usage Notes

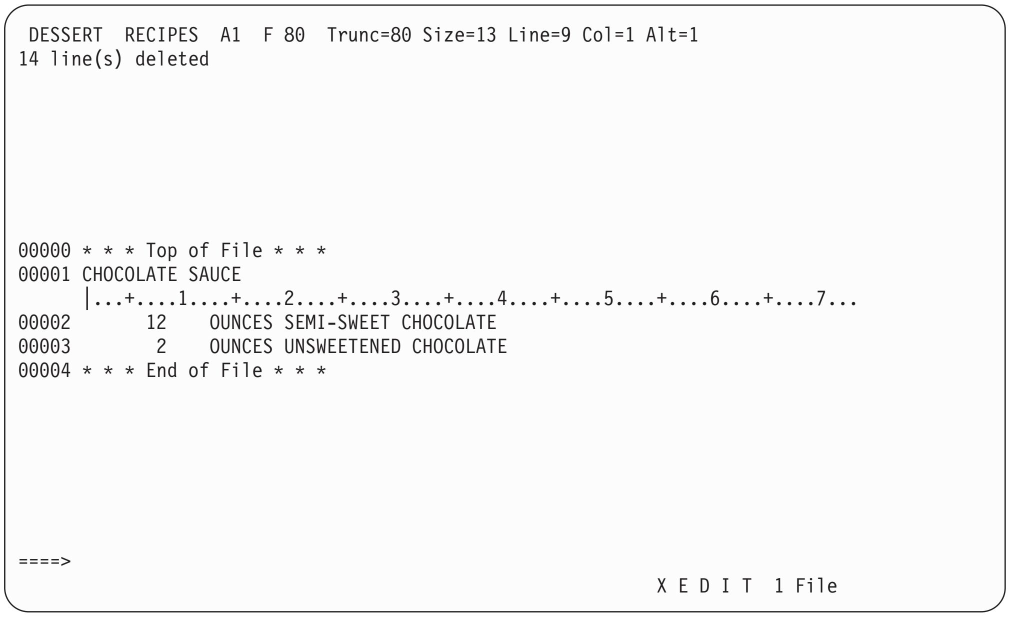
1. You can clear a line by (1) pressing the spacebar once in column one and then pressing ERASE EOF, or (2) repeatedly pressing the DELETE key. If a line is cleared in this manner, a blank line remains in the file. If you do not press the spacebar, the data comes back in the line the next time you press ENTER. This prevents you from erasing an entire line accidentally.
2. If the current line is the Top of File line, this is counted as one of the deleted lines.

Examples

Figure 7 on page 80 is a before-and-after example of the DELETE subcommand.



```
 DESSERT  RECIPES  A1  F 80  Trunc=80 Size=19 Line=9 Col=1 Alt=0

00000 * * * Top of File * * *
00001 CREAM PUFFS
00002        2    OUNCES BUTTER
00003        1/2  TEASPOON SUGAR
00004        1/2  CUP FLOUR
00005             PINCH OF SALT
00006        2    EGGS
00007        2    CUPS HEAVY CREAM, WHIPPED
00008
00009  VINAIGRETTE SAUCE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00010        1/2  CUP OLIVE OIL
00011        1/2  CUP VINEGAR
00012             PINCH OF SALT
00013             DASH OF PEPPER
00014
00015 CHOCOLATE SAUCE
00016       12    OUNCES SEMI-SWEET CHOCOLATE
00017        2    OUNCES UNSWEETENED CHOCOLATE
====> delete /chocolate/
                                                        X E D I T  1 File
```



```
 DESSERT  RECIPES  A1  F 80  Trunc=80 Size=13 Line=9 Col=1 Alt=1
14 line(s) deleted









00000 * * * Top of File * * *
00001 CHOCOLATE SAUCE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00002       12    OUNCES SEMI-SWEET CHOCOLATE
00003        2    OUNCES UNSWEETENED CHOCOLATE
00004 * * * End of File * * *









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

Figure 7. DELETE Subcommand — Before and After

Responses

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

501I *nn* line(s) deleted

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

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

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

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

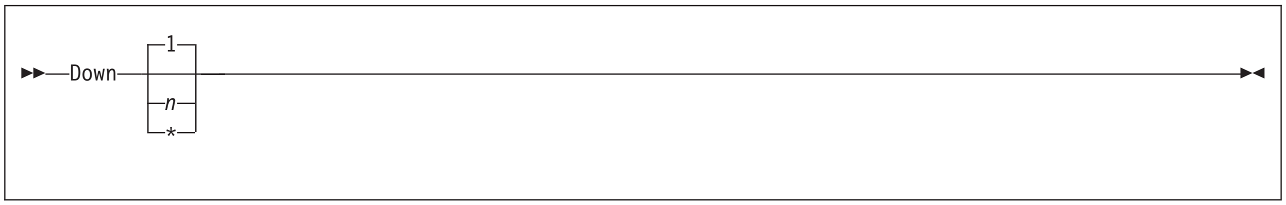
Messages and Return Codes

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

where return codes are:

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

DOWN



Purpose

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.)

Operands

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 *n* is not specified, the pointer moves down only one line.

Usage Notes

1. The DOWN *n* subcommand is equivalent to a plus (+) target definition.

   For example:

   ```
   down 3
   ```

   is equivalent to

   ```
   +3
   ```

Examples

See the example of the NEXT subcommand on page 169.

Responses

The line pointed to becomes the new current line.

If you issue the DOWN subcommand while editing a file containing no records, the following message is displayed:

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

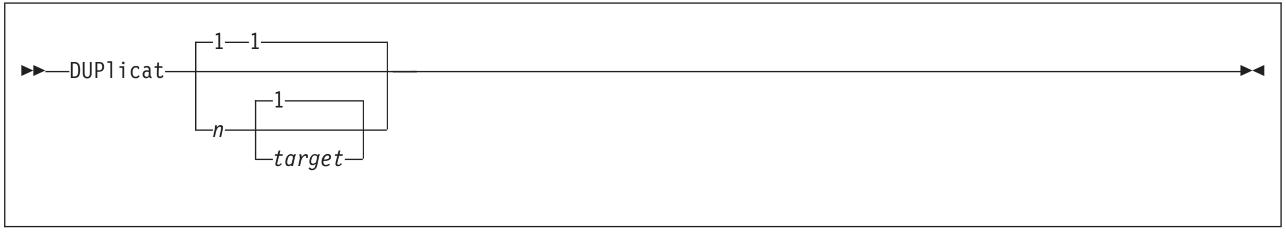
Messages and Return Codes

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

where return codes are:

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

DUPLICAT



Purpose

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).

Operands

n specifies the number of times the line(s) is to be duplicated. If *n* is not specified, 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 *target* is not specified, only the current line is duplicated.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see “LOCATE” on page 147 and *z/VM: XEDIT User’s Guide*.

Responses

The last line duplicated becomes the new current line.

Messages and Return Codes

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

where return codes are:

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

EMSG

```
           (1)
>>──EMSG──┬──────────────────────────┬──────────────────────────────────────────><
          └─┬──────────┬──┬────────┬─┘
            ├─mmmnnns──┤  └─text───┘
            └─mmmnnnns─┘
```

Notes:

1 If *text* is not specified, a blank message is displayed.

Purpose

The EMSG subcommand displays a message at the terminal and sounds the alarm. You can use it in macros and modules that interface with XEDIT and whose messages follow the rules for z/VM messages (see the appropriate messages book for details). The severity of the message determines whether the alarm is sounded.

Operands

text
is the text of the message to be displayed. If *text* is not specified, a blank message is displayed.

mmmnnns
mmmnnnns
is the message identification.

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

nnn **or** *nnnn*
is the three- or four-digit message number. You can use it to find additional information in the appropriate messages book.

s indicates the severity and is one of the following:

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

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

The message is prefixed by DMS before being displayed.

Usage Notes

1. The SOS ALARM subcommand can sound the alarm without displaying any message.
2. Messages are displayed according to the SET MSGMODE setting (ON or OFF).
3. The message identification is processed according to the CP EMSG setting and CMS processing by severity (see *z/VM: CP Commands and Utilities Reference*).

However, messages with the severity code of R also respect the CP EMSG setting. In addition, the message is processed according to the SET MSGMODE setting (LONG or SHORT).

4. The message identification must not contain imbedded blanks. It must be preceded by only one blank delimiter.
5. The EMSG subcommand can also display messages on a typewriter terminal.
6. If you are in a GUI environment, messages that do not fit in the message area are displayed in the XEDIT Messages window.

Responses

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

Messages and Return Codes

This subcommand issues no messages, but it issues return codes as follows:

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

EXPAND

```
         ┌─1──────┐
►►──EXPand──┼────────┼──────────────────────────────────────────►◄
         ├─target─┤
         └─*──────┘
```

Purpose

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 with 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.

Operands

target
defines the number of lines to be expanded. Lines are expanded starting with the current line and continuing up to, but not including, the target line. If *target* is not specified, only the current line is expanded.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

*****
the rest of the file is expanded.

Usage Notes

1. For EXPAND to work properly, lines containing backspace characters (X'16') must have been entered into the file with SET IMAGE CANON in effect, that is, compound characters ordered with backspaces arranged singly between the characters that overlay each other.
2. See "COMPRESS" on page 62 for an explanation of how to use the COMPRESS, EXPAND, and SET TABS subcommands to reposition data.
3. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.

Examples

See the "Examples" section of the COMPRESS subcommand on page 62.

Responses

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

1. Is unchanged if SET STAY ON is in effect
2. Moves to the last line expanded, if SET STAY OFF is in effect (the default)

Messages and Return Codes

504E *nn* **line(s) {truncated|spilled} [RC=3]**

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

where return codes are:

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

EXTRACT

```
            ┌───────────┐
            ▼           │
►►──EXTract────/operand─┴──┬───┬──────────────────────────────────►◄
                           └─/─┘
```

Purpose

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

Operands

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

operand
may be any one of the following keywords. A complete description of each operand follows. For additional information on some of the operands, refer to its corresponding SET subcommand (if applicable).

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

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

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

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

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

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

APL
: returns ON or OFF as defined in the SET APL subcommand.

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

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

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

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

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

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

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

Note: stBASEFT.1 is blank if the file that was loaded was a BFS (byte file system) file.

BFSLine
: returns the values used by XEDIT to translate a byte file system (BFS) byte stream into records.

```
BFSLINE.0     number of variables returned
       .1     [lrecl|NL|CRLF|CRNL|/string/|/hexstring/]
```

BRKkey
returns ON or OFF as the CP terminal brkkey setting defines. If ON, then also returns the key.

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

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

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

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

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

COLOR

```
►►──EXTract──/COLOR──┬─field─┬──┬───┬────────────────────────────────►◄
                     └─*─────┘  └─/─┘
```

COLOR *field*

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

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

COLOR *

returns the areas of the screen and their respective colors, extended highlighting, highlighting, and programmed symbol set options as they have been specified in the SET COLOR subcommand. Note, this operand may be specified as COLOR or COLOUR.

```
 COLOR.0      number of variables returned

      .1      ARROW     color exthi HIGH|NOHIGH PSs

      .2      CMDLINE   color exthi HIGH|NOHIGH PSs

      .3      CURLINE   color exthi HIGH|NOHIGH PSs

      .4      FILEAREA  color exthi HIGH|NOHIGH PSs

      .5      IDLINE    color exthi HIGH|NOHIGH PSs

      .6      MSGLINE   color exthi HIGH|NOHIGH PSs

      .7      PENDING   color exthi HIGH|NOHIGH PSs
```

```
         .8       PREFIX    color exthi HIGH|NOHIGH PSs

         .9       SCALE     color exthi HIGH|NOHIGH PSs

         .10      SHADOW    color exthi HIGH|NOHIGH PSs

         .11      STATAREA  color exthi HIGH|NOHIGH PSs

         .12      TABLINE   color exthi HIGH|NOHIGH PSs

         .13      TOFEOF    color exthi HIGH|NOHIGH PSs
```

COLPtr

returns ON or OFF as defined in the SET COLPTR subcommand.

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

COLumn

returns the column number of the column pointer.

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

CTLchar

```
►►──EXTract──/CTLchar──┬──────┬──┬───┬──────────────────────────────►◄
                       └─char─┘  └─/─┘
```

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

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

CTLchar *char*

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

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

CURLine

If you are using a display terminal, EXTRACT /CURLINE/ returns the line number of the current line as specified in the SET CURLINE subcommand, the contents of the current line from column one through the truncation column (excluding trailing blanks), and whether it has been changed or inserted during this editing session.

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

If you are using a typewriter terminal, EXTRACT /CURLINE/ returns the contents of the current line from column one through the truncation column (excluding trailing blanks), and whether it has been changed or inserted during this editing session.

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

CURSor

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

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

DISPlay

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

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

EDIRName

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

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

EFMode

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

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

Note: EFMODE.1 is blank if the file that was loaded was a byte file system (BFS) file.

EFName

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

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

Note: EFNAME.1 is blank if the file that was loaded was a byte file system (BFS) file.

EFType

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

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

Note: EFTYPE.1 is blank if the file that was loaded was a byte file system (BFS) file.

ENTer

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

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

EOF

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

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

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

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

EPName
returns the byte file system (BFS) path name at the time the file was loaded.

Depending upon the BFS path name format used when the file was loaded, this can be a relative path name, an absolute path name, or a fully qualified path name. See “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the different forms of the BFS path name. EPNAME.0=0 if the file that was loaded was not a BFS file.

```
EPNAME.0      number of variables returned
      .1      number of characters in EPNAME.2
      .2      pathname
```

ESCape
returns ON or OFF and the escape character defined in the SET ESCAPE subcommand.

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

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

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

ETMODE
returns ON or OFF as specified in the SET ETMODE subcommand.

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

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

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

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

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

FMode
returns the two-character file mode.

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

Note: FMODE.1 is blank if the file originally edited was a BFS file and SET FMODE has not been entered.

FName
returns the eight-character file name.

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

Note: FNAME.1 is blank if the file originally edited was a BFS file and SET FNAME has not been entered.

FType
returns the eight-character file type.

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

Note: FTYPE.1 is blank if the file originally edited was a BFS file and SET FTYPE has not been entered.

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

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

GUI
returns ON or OFF to indicate whether the ring is being displayed in a GUI window rather than a terminal emulator screen.

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

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

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

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

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

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

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

INPmode
returns ON or OFF as determined by whether you are in input mode.

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

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

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

LASTmsg
returns the last message the editor issued. (Depending on the SET MSGMODE operands specified, this message may or may not have been displayed.) If CP SET EMSG OFF is in effect, LASTMSG is not updated, even if SET MSGMODE ON. Severe (S) and Terminating (T) messages are handled as if CP EMSG ON is in effect.

The updating of the LASTMSG buffer is handled differently for the message COUNT issues containing the number of occurrences of a string. See “COUNT” on page 68 for more information.

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

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

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

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

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

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

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

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

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

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

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

LOCk
returns ON or OFF to indicate whether the editor locked the file at the time it was first loaded. ON is returned even if the file system has since dropped the lock.

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

LRecl
returns the logical record length of the file.

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

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

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

MACRO
returns ON or OFF as specified in the SET MACRO subcommand.

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

MASK

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

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

MEMber

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

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

MSGLine

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

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

MSGMode

returns ON or OFF and LONG or SHORT as defined in the SET MSGMODE subcommand.

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

NAMetype

returns `CMS` or `BFS` as defined by the current NAMETYPE value.

```
NAMETYPE.0    number of variables returned
       .1    CMS or BFS
```

NBFile

returns the number of files you are currently editing.

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

NBScope

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

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

NONDisp

returns the character defined in the SET NONDISP subcommand.

NONDISP.0	number of variables returned
.1	nondisp character

NULls

returns ON or OFF as specified in the SET NULLS subcommand.

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

NUMber

returns ON or OFF as specified in the SET NUMBER subcommand.

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

PA

```
                  ┌─*─┐
►►──EXTract──/PA──┼───┼─────┬───┬───────────────────────────────────────►◄
                  └─n─┘     └─/─┘
```

PA*n*

returns BEFORE, AFTER, ONLY, or IGNORE and the PA*n* key definition as specified in the SET PA*n* subcommand. If the PA*n* key is undefined, PAn.0=0 and no other variables are returned.

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

PA*

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

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

PACK

returns ON or OFF as specified in the SET PACK subcommand.

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

PENDing

```
►►──EXTract──/PENDing──┬───────┬──┬─────────┬──┬─name─┬──────────────────►
                       └─BLOCK─┘  └─OLDNAME─┘  └─*────┘

   ┌─:1──*──────────────────────┐
►──┼────────────────────────────┼──┬───┬────────────────────────────────►◄
   └─target1──┬─*───────┬───────┘  └─/─┘
              └─target2─┘
```

returns information from the *pending list* (see “SET PENDING” on page 304) with respect to a particular subcommand or macro name or the first pending entry.

BLOCK

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

OLDNAME

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

name

indicates the name of the prefix subcommand or macro for which you are searching. If OLDNAME is also specified, *name* must be the original name of the prefix subcommand or macro, regardless of whether a synonym has been assigned to that name. Otherwise, it is assumed to be a synonym (that is, a new name) or a name without a synonym.

*

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

target1 target2

indicates the range in the file where the associated prefix subcommand or macro must be located. If only *target1* is specified, the search starts at *target1* and runs to the end of the file. *Target1* is obtained relative to the top of the file. For example, EXTRACT /PENDING * +3 / starts at the top of the file and goes 3 lines forward in the file to begin the search. *Target2* is obtained relative to *target1*. If *target1* is line 3 and you enter a +5 as *target2*, the search is from line 3 through line 8, inclusive of both target lines. If *target2* is specified prior to *target1*, the result is the same as if you had specified *target1 target2*. For example, EXTRACT /PENDING * :10 :2/ produces the same results as EXTRACT /PENDING * :2 :10/. If no targets are specified, the entire pending list is searched, that is, the entire file is searched, starting from the top of the file (:1).

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

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

PF

```
            ┌─*─┐
►►──EXTract──/PF──┼───┼──┬───┬──────────────────────────────────────►◄
            └─n─┘  └─/─┘
```

PF*n*

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

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

PF*

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

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

PName

returns the current active byte file system (BFS) path name. The total BFS path name can be up to 1023 characters in length.

Depending upon the path name format used on SET PNAME or when the file was loaded, this can be a relative path name, an absolute path name, or a fully qualified path name. See “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the different forms of the BFS path name. PNAME.0=0 if the file originally edited was a CMS file and SET PNAME has not been entered.

```
PNAME.0      number of variables returned
     .1      number of characters in PNAME.2
     .2      BFS path name
```

Point

```
►►──EXTract──/Point──┬───┬──┬───┬─────────────────────────────────────►◄
                     └─*─┘  └─/─┘
```

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

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

Point *

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

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

PREfix

```
►►──EXTract──/PREfix──┬──────────────────────┬──┬───┬──────────────────────►◄
                      └─Synonym──┬─name─┬─┘  └─/─┘
                                 └─*────┘
```

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

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

PREfix Synonym *name*

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

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

PREfix Synonym *

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

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

RANge

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

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

RECFm

returns the record format, F, V, FP, or VP, defined in the SET RECFM subcommand.

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

REMOte

returns ON or OFF depending upon whether a remote terminal is being used or upon the setting defined in the SET REMOTE subcommand.

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

RESERved

```
►►──EXTract──/RESERved──┬───┬──┬───┬─────────────────────────────────►◄
                        └─*─┘  └─/─┘
```

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

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

RESERved *

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

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

RING

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

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

SCALe

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

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

SCOPE
returns DISPLAY or ALL as specified in the SET SCOPE subcommand.

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

SCReen
returns the attributes of the screens as defined in the SET SCREEN subcommand.

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

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

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

Seq8
returns OFF if the XEDIT command or LOAD subcommand was issued with the NOSEQ8 operand; if not, returns ON.

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

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

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

SHADow
returns ON or OFF as specified in the SET SHADOW subcommand.

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

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

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

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

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

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

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

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

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

STAY

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

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

STReam

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

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

SYNonym

```
►►──EXTract──/SYNonym──┬──────┬──┬───┬──────────────────────────────►◄
                       ├─name─┤  └─/─┘
                       └─*────┘
```

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

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

SYNonym *name*

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

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

SYNonym *

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

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

TABLine

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

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

TABS

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

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

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

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

If the last target located was specified with *&*, only information about the last string found is returned. For example, if the last target was located with the following command:

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

and the line located,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

TOL

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

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

TRANSLat

returns ON or OFF, depending on whether the user has defined pairs of uppercase translate characters using the SET TRANSLAT subcommand.

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

TRunc

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

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

UNIQueid

returns the unique identifier associated with the file. The identifier has the form *rrrnnnnn* where *rrr* is the number XEDIT associates with the ring and *nnnnn* is the current autosave number. Note, when the ring number, *rrr*, is less than 100, leading zeros are dropped. The UNIQUEID is also the file name for the AUTOSAVE file.

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

UNTil

returns the file type through which updates have been applied as specified in the XEDIT command or LOAD subcommand.

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

UPDate

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

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

VARblank

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

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

Verify

returns H (if SET VERIFY with the HEX option was previously issued), the verification columns, and ON or OFF as specified in the SET VERIFY subcommand.

```
VERIFY.0      number of variables returned
     .1      ON|OFF
     .2      [H]startcol endcol [...[H]startcol endcol]
```

See “Example 6” on page 109.

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

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

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

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

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

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

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

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

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

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

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

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

Notes for Macro Writers

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

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

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

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

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

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

   If XEDIT is invoked from an EXEC or XEDIT macro, that EXEC or macro is *active* until you exit from XEDIT and subsequently from the EXEC or macro. Issuing EXTRACT from the XEDIT command line may set variables in the EXEC or macro that invoked XEDIT.
4. EXTRACT/MASK/ issued from an EXEC 2 environment always results in return code 16 since the MASK has 256 characters. To obtain the same information, you can use TRANSFER MASK. Issuing EXTRACT/MASK/ from a REXX environment returns the mask properly, because REXX imposes no length restrictions.
5. If SET HEX ON is in effect, you can specify targets in hexadecimal.
6. Some EXTRACT operands return a value that reflects how the screen would look if it were displayed at the time that EXTRACT is being executed. The value returned reflects relative changes due to the execution of prefix subcommands and macros. It does not necessarily reflect the final content of the screen, which is eventually displayed.

Examples

The following are examples using the EXTRACT subcommand.

Example 1

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

```
set lrecl 65
```

and then execute an XEDIT macro that issues:

```
extract /ACTION/
```

The following variables are set:

```
        ACTION.0 = 1
        ACTION.1 = ON
```

Example 2

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

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

The following variables are set:

```
        COLUMN.0 = 1
        COLUMN.1 = 1
        FNAME.0  = 1
        FNAME.1  = SAMPLE
        PF3.0    = 2
        PF3.1    = BEFORE
        PF3.2    = QUIT
```

Example 3

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

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

The following variables are set:

```
         PENDING.0 = 7
         PENDING.1 = 6
         PENDING.2 = WW
         PENDING.3 = WW
         PENDING.4 = BLOCK
         PENDING.5 = 3
         PENDING.6 = 57
         PENDING.7 = null
```

Example 4

While editing a file you execute an XEDIT macro that issues:

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

The following variables are set:

```
            CASE.0 = 2
            CASE.1 = MIXED
            CASE.2 = RESPECT
         EXTRACT.0 = 1
         EXTRACT.1 = COLOR ALL
```

Note: The return code is 5 because COLOR ALL is an invalid operand and nothing is returned for AUTOSAVE (see note 2 on page 107).

Example 5

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

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

With the line you just added as the current line, you then execute an XEDIT macro that issues:

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

The following variables are set:

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

Example 6

If you issue:

```
SET VERIFY 1 10 H 11 20 21 30 H 31 40
```

40 columns of data are displayed: 1-10 and 21-30 in EBCDIC and 11-20 and 31-40 in hexadecimal. If you then issue:

```
EXTRACT /VERIFY/
```

The VERIFY.2 variable is set to contain pairs of columns. A column range displayed in hexadecimal is indicated by the letter H that prefixes the starting column number.

```
VERIFY.2 = 1 10 H11 20 21 30 H31 40
```

There is no blank between the H and the starting column number.

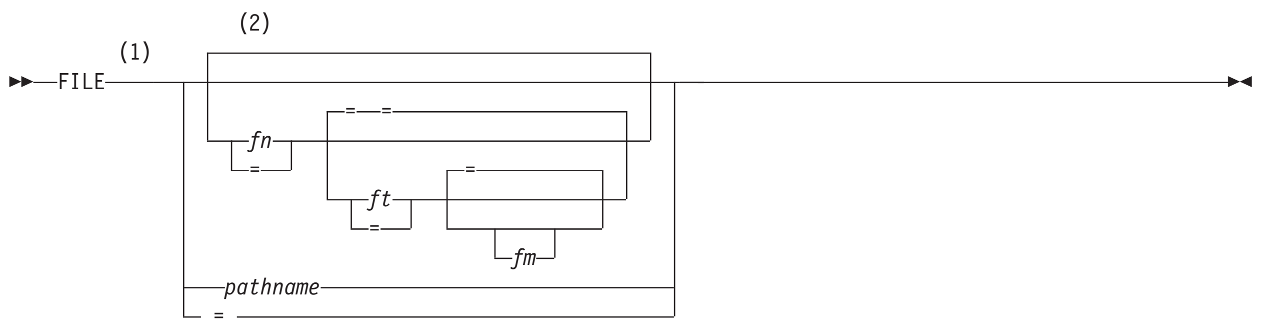
Messages and Return Codes

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

where return codes are:

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

FILE



Notes:

1 The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the byte file system (BFS) format required for BFS files (*pathname*).

2 If file ID is not specified, the default file ID is = = = (for NAMETYPE CMS) or = (for NAMETYPE BFS).

Purpose

Use the FILE subcommand or FFILE to write the edited file to a minidisk, a Shared File System (SFS) directory, or a byte file system (BFS) regular file. You can also use the FILE subcommand to change the current file identifier.

Operands

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

ft indicates the file type for the SFS or minidisk file.

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

pathname
is the name of the byte file system (BFS) file to be written. See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name.

The length of the path name entered on this subcommand is limited. Use `SET PNAME` *`pathname_fragment1`* (or `PN` *`pathname_fragment1`*) and `SET PNAME APPEND` *`pathname_fragment2`* one or more times. Then use `FILE =` if you wish to use a longer path name. Refer to Chapter 1, "Rules and Conventions," on page 1 for more information about the maximum lengths allowed for XEDIT subcommands.

Usage Notes

1. To use the FILE subcommand to write a file to a minidisk, you must have the minidisk accessed in read/write mode.
2. To use the FILE subcommand to *change* another user's SFS file, you must be properly authorized:

- For a file in a directory control directory, you must have directory control write authority to the directory, and you must access the directory in read/write mode.

  When you access another user's directory, it is, by default, accessed in read-only mode. To access another user's directory in read/write mode, specify the FORCERW option on the ACCESS command.
- For a file in a file control directory, you must have write authority to the file; the directory can be accessed in either read/write or read-only mode. However, if the directory is accessed in read-only mode and the CMS command SET RORESPECT ON has been issued, an attempt to write will fail.

3. To use the FILE subcommand to *create* a file in another user's SFS directory, you must be properly authorized:
   - For directory control directories, you need directory control write authority to the directory, and you must access the directory in read/write mode. When the file is created, the owner of the directory becomes the owner of the new file, but because you have directory control write authority, you will be able to write to the file.

     When you access another user's directory, it is, by default, accessed in read-only mode. To access another user's directory in read/write mode, specify the FORCERW option on the ACCESS command.
   - For file control directories, you need write authority to the user's directory. In this case, the directory can be accessed in either read/write or read-only mode. When the file is created, the owner of the directory becomes the owner of the new file, but because you created the file, you have write authority for the file. However, if the directory is accessed in read-only mode and the CMS command SET RORESPECT ON has been issued, an attempt to write will fail.
4. To use the FILE subcommand to change a BFS file, you must have permission to write to the file. To create a file in a BFS, you must have permission to write to the parent directory.
5. Permissions for a new BFS file are set based on the current value of the mask, with the exception of execute permissions, which are not set to ON. Use OPENVM PERMIT to change the defaults. See *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM for more information on these OPENVM commands.
6. The BFSLINE value determines whether end-of-line characters are inserted in the file. The RECFM value determines whether or not trailing blanks are stripped from the file. Refer to "SET BFSLINE" on page 245 and "SET RECFM" on page 319 for more information.
7. Use the QUERY DIRATTR command or the QUERY ACCESSED command to determine whether the SFS directory has the FILECONTROL or the DIRCONTROL attribute. (In the QUERY ACCESSED display, the *Vdev* column contains *DIR* for FILECONTROL directories and *DIRC* for DIRCONTROL directories.)
8. You can change the file name, file type, and file mode during an editing session by using the SET FNAME, SET FTYPE, and SET FMODE subcommands (as well as by specifying a different file identifier on the FILE subcommand).

   You can change the path name during an editing session by using the SET PNAME subcommand (or you can specify a different path name on the FILE subcommand).

9. If you change the file identifier (to a unique file identifier) and the file being edited has previously been written to disk, SFS file space, or a BFS, the copy with the original name of the file is not altered.
10. When NAMETYPE CMS is in effect, specifying an equal sign (=) for the *fn, ft,* and *fm* operands means the corresponding FNAME, FTYPE, and FMODE values are used. If not specified as an equal sign, the operand will be translated to uppercase.

    When NAMETYPE BFS is in effect, specifying an equal sign (=) for the *pathname* operand means the value set using SET PNAME will be used. The path name is converted to uppercase only when SET CASE UPPER is in effect.

    If you change the NAMETYPE value and then attempt to use an equal sign without setting the appropriate FNAME, FTYPE, or PNAME values, the editor returns an error. If you change the NAMETYPE value to CMS and then use an equal sign without setting FMODE, the mode defaults to A1.
11. Commit behavior of BFS files is different from that of SFS or minidisk files. For example, if the editor encounters an error when writing to a minidisk or SFS file, your changes are reversed and the original file is preserved. This is not the case for BFS files. The editor creates a copy of the file in XEDTEMP CMSUT1 on your A disk so you can recover your data if an error occurs when writing the file to the byte file system.

    If the editor is unable to create the XEDTEMP CMSUT1 file, you will receive a message describing the problem, and a second message:

    ```
    595E  Not able to create CMS file used for recovery.  Correct
          error or QQUIT to exit without writing file
    ```

    If the editor encounters an error while writing the byte file system file such that the contents of an existing file are damaged, you will receive a message describing the problem, and a second message to tell you that you must take action to recover your file.

    ```
    024E  File XEDTEMP CMSUT1 A1 contains file contents; use
          OPENVM PUT to recover file
    ```
12. FILE is defined as a synonym to PFILE (the protected equivalent of FILE). The PFILE subcommand issues the:

    ```
    594E File {fn ft fm|pathname} already exists
         or changed; use FFILE or SSAVE
    ```

    message if:
    - the *disk* file has been changed outside this editing session. For example, if you were editing the same file in two different rings, changed the file in the second ring, filed it, changed the file in the first ring and attempted to file that one, the 594E message would be displayed.
    - the file identifier was changed so it is identical to that of an *existing* file.

    If you are sure you want to save the file you are editing, issue FFILE (unprotected FILE, abbreviated FF) to save the file and overlay any existing version. FFILE is a synonym for COMMAND FILE.
13. If you try to FILE a shared file that another user has modified during your editing session, the editor stops the FILE operation and displays message 594E.

    ```
    594E File {fn ft fm|pathname} already exists or changed;
         use FFILE or SSAVE
    ```

    At this point you can decide whether to write over the other user's changes or preserve them. If you want to write over them, deleting the changes, enter

FFILE. If you want to preserve the other user's changes, reissue the FILE subcommand using a unique file identifier to save the changes under a different name. Later, you can determine the differences between the two versions of the file and combine them. Note, the LOCK option of the XEDIT command prevents other users from changing an SFS file control file while you are editing it.

14. If you change the member name (to a unique member name) and the member has previously been written to a disk or directory, the copy with the original name of the member is not altered.

    However, if you change the member name while editing a member so the new member name is the same as that of an existing member in that library, the editor stops the FILE operation and displays the following message:

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

    If you want to write over the existing member, enter FFILE (abbreviated as FF). If not, change the member name so it is unique and reissue the FILE subcommand.

15. Members with identical names contained in libraries with identical names, but residing on different disks or directories, are not written over if you attempt to change the file mode. The following warning is issued:

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

    For example:

    ```
    PROJA MACLIB A1               PROJA MACLIB B1

    ENTER                         ENTER
    LEAVE                         LEAVE
    FORMAT                        FORMAT
    COMPUTE                       COMPUTE
    ```

    If you are editing the ENTER member in the PROJA MACLIB that resides on your file mode B and attempt to change the file mode by issuing FILE = = A, the warning message is issued.

16. If you try to FILE an empty file, you will get the following message:

    ```
    595E Issue SSAVE/FFILE to a directory to write an empty file
         or QQUIT to exit without writing file
    ```

    and remain in XEDIT.

    SSAVE/FFILE will only succeed for empty files which reside in an SFS or BFS directory. In addition, since at least one record is required for a packed file, empty packed files are not supported. Empty autosave files are not written. Empty MACLIB members are also unsupported. When in update mode, any update that deletes all the records in the merged source file is not allowed.

    For any unsupported case, you will receive the following error message:

    ```
    559E Empty file fn ft fm not written
    ```

    and remain in XEDIT.

17. Members of a MACLIB must have a file type of MEMBER.

18. When editing two or more members of a MACLIB in member mode, the lock on the library is not deleted until the last member that had the lock option in effect exits the ring.

19. When editing a file in update mode, the file written will contain changes to the file defined by the latest update level applied plus any changes you made while editing. The original source file to which the update(s) were applied is *not* changed. If the updates were merged (see the MERGE option on the XEDIT

command on page 16), the file written will contain changes defined by all update levels plus any changes you made.

If the latest update level contained update control comment records, they will be placed at the beginning of the file written, regardless of their original position in the update file. For more information on the CMS UPDATE command, see *z/VM: CMS Commands and Utilities Reference*.

20. The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the BFS format required for BFS files (*pathname*). NAMETYPE may be set using an XEDIT option or by the SET NAMETYPE subcommand.
21. When you XEDIT a BFS file and attempt to write the file when the record format is V (variable), you will see an error message if the line length exceeds 65 535. (The maximum record length for a variable file in the CMS record file system is 65 535.)
22. When a new BFS file is created, the owning UID established is the effective UID of the VM User ID on which the request was issued. The GID is the GID of the parent directory. Use OPENVM OWNER to change the defaults. Refer to *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM OWNER for more information.

Notes for Macro Writers

1. If FILE is issued from a macro, the file is written to disk or directory, but control remains in the macro. When the macro finishes executing, control is returned to the editor.

   If multiple files were being edited, only the current file is written to disk or directory and removed from the set of files being edited.

   If only one file was being edited, a FILE issued from a macro sets the return code to 1 and executes the FILE when the macro completes execution. After issuing the FILE, any subcommands issued in the macro result in a return code of 6.
2. You cannot issue FILE, FFILE or PFILE from a prefix macro. However, if you issue a FILE, which consists of a SAVE and a QUIT, the SAVE is performed and you receive error message 509E on the QUIT.

Responses

If only one file was edited, the CMS ready message indicates the file has been written to disk or directory. The user is returned to the environment that invoked the editor.

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

On a typewriter terminal, the following message is displayed:

```
553I Editing file: {fn ft fm|pathname}
```

Messages and Return Codes

002E File *fn ft fm* **not found [RC=28]**
007E File *fn ft fm* **is not fixed, 80-character records [RC=32]**
024E File XEDTEMP CMSUT1 A1 contains file contents; use OPENVM PUT to recover file
033E File [*fn ft fm***] is not a {library|regular BFS file} [RC=32]**
037E Base file for *fn ft fm* **is in a DIRCONTROL directory accessed read/only [RC=12]**

037E	**Filemode** *mode* **is accessed as read/only [RC=12]**
039E	**No entries in library** *fn ft fm* **[RC=32]**
048E	**Invalid filemode** *mode* **[RC=24]**
054E	**Incomplete [or incorrect] fileid specified [RC=24]**
062E	**Invalid character in fileid {***fn ft fm***|***pathname***} [RC=20]**
069E	**Filemode** *mode* **not accessed [RC=36]**
104S	**Error** *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, 99, or 100]**
105S	**Error** *nn* **writing file** *fn ft fm* **on disk or directory [RC=55, 70, 76, 99, or 100]**
157S	**MACLIB limit exceeded [RC=88]**
167S	**Previous MACLIB function not finished [RC=88]**
229E	**Unsupported OS dataset, error** *nn* **[RC=80, 81, 82, 83, or 84]**
509E	*Subcommand* **subcommand not valid from a prefix macro [RC=4]**
512E	**This is not allowed in CMS subset mode [RC=100]**
520E	**Invalid operand:** *operand* **[RC=5]**
531E	**BFS file space is full; clear some space [RC=13]**
531E	**Disk or file space is full; set new filemode or clear some space [RC=13]**
532E	**Disk or file space is full; AUTOSAVE failed [RC=13]**
554E	**Not enough virtual storage available [RC=104]**
559E	**Empty file** *fn ft fm* **not written [RC=88]**
560E	**Not enough space for serialization between TRUNC and LRECL**
579E	**Records truncated to** *nn* **when added to** *fn ft fm* **[RC=3]**
594E	**File {***fn ft fm***|***pathname***} already exists or changed; use FFILE or SSAVE [RC=3]**
595E	**Issue SSAVE/FFILE [to a directory] to write an empty file or QQUIT to exit without writing file [RC=88]**
595E	**Not able to create CMS file used for recovery. Correct error or QQUIT to exit without writing file**
595E	**Not able to obtain lock. Issue SSAVE/FFILE to write file without locking or QQUIT to exit without writing file [RC=70]**
598S	**Unable to build update file: internal list destroyed [RC=7]**
599S	**Unable to build update file: serialization destroyed [RC=7]**
622E	**Insufficient free storage for reading map [RC=104]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code** *rc* **and reason code** *rs* **[RC=55, 104]**
1138E	**File sharing conflict for file {***fn ft fm***|***pathname***} [RC=70]**
1141W	**User filespace threshold exceeded**
1184E	**A directory is not found, or you are not permitted to use a directory in** *pathname* **[RC=28]**
1215E	**File** *fn ft fm* **is locked or in use by another user [RC=70]**
1258E	**Not {authorized|permitted} to write file {***fn ft fm***|***pathname***} [RC=12]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=55, 70, 76, 99, or 100]**
1301S	**Rollback error** *nn***, file** *fn ft fm* **left open [RC=100]**
1350E	**AUTOSAVE must be targeted to your own directory [RC=12]**
2105E	**Permission is denied [RC=24]**
2120E	**Unable to resolve path name** *pathname* **[RC=24]**
2131E	**[FNAME] [FTYPE] [PNAME] {is|are} not set and NAMETYPE {CMS|BFS} is in effect [RC=24]**
2134E	**Return code** *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname***] [RC=100]**
2154E	**File {***fn ft fm***|***pathname***} is migrated and implicit RECALL is set to OFF [RC=50]**

2155E DFSMS/VM error occurred during creation or recall of file {*fn ft fm*|*pathname*} [RC=51]

2526E File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code *acs rcode* **[RC=51]**

where return codes are:

0	Normal
1	File has been filed and was the only one edited
3	File already exists; truncated or spilled
4	Invalid when issued from a prefix macro
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
7	Error building the update file
12	Minidisk defined in file mode is read-only; or not authorized to write to file; or cannot AUTOSAVE to another user's directory or you do not have permission for the BFS file or byte file system mounted read-only
13	Minidisk or file space or byte file system is full
20	Character in file name or file type or path name not valid
24	File mode or path name not valid
28	File not found
31	A rollback occurred
32	File is not a library, or library has no entries, or file is not fixed, 80 char. records or BFS file is not a regular file, or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
84	Unsupported OS data set
88	Previous MACLIB function not finished, MACLIB limit exceeded, or unsupported attempt to write an empty file
99	A required system resource is not available
100	Error *nn* {reading|writing} file {from|to} disk or directory or byte file system
104	Insufficient storage available

FIND

```
►►──Find──text──────────────────────────────────────────────────────────────►◄
```

Purpose

Use the FIND subcommand to search forward in the file for the first line that starts with the text specified in the operand. Only nonblank characters in the operand are checked against the file. The search starts with the line after the current line.

Operands

text
: is any text you expect to find beginning in column one of the next file line. If *text* contains imbedded blanks, those character positions in the file line are not checked.

Usage Notes

1. If SET IMAGE ON is in effect, tab characters are converted to blanks (or filler characters) before the search, and the search starts in the first tab column.
2. To represent a blank in the operand, use an underscore character (_).
3. Use only one blank as a delimiter after the FIND subcommand; the operand starts after the blank.
4. If you want to do the same type of search backward, see “FINDUP” on page 120. See “NFIND” on page 172 and “NFINDUP” on page 173 to do searches which do not start with *text*.
5. If SET WRAP OFF is in effect, the search continues until the end of file.

   If SET WRAP ON is in effect, the search continues through the entire file and stops when the line where it started is reached again. When a wrap occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```
6. The FIND subcommand updates the LASTLORC buffer. See “SET LASTLORC” on page 281.

Examples

In the following example, the FIND subcommand is used to search for the first occurrence of Ding in the text.

Current Line:

```
===== Dingoes are wild dogs of Australia.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====         .
=====         .
===== They are called Dingoes.
=====         .
=====         .
=====         .
===== Dingoes do not bark.

find Ding        (text must start in column one of a line)
```

New Current Line:

```
===== Dingoes do not bark.
```

Responses

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

Messages and Return Codes

545E Missing operand(s) [RC=5]
586E Not found [RC=2]
592W Wrapped ....

where return codes are:

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

FINDUP

```
►►──┬─FINDUp─┬──text──────────────────────────────────────────────────►◄
    └─FUp────┘
```

Purpose

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 nonblank characters in the operand are checked against the file. The search starts with the line before the current line.

Operands

text
: is any text you expect to find beginning in column one of a file line that appears before the current line. If *text* contains imbedded blanks, the corresponding character positions in the file are not checked.

Usage Notes

1. If the SET IMAGE ON subcommand is in effect, tab characters are converted to blanks (or filler characters) before the search, and the search starts in the first tab column.
2. To represent a blank in the operand, use an underscore character (_).
3. Use only one blank as a delimiter after the subcommand; the operand starts after the blank.
4. If you want to do the same type of search forward, see "FIND" on page 118. See "NFIND" on page 172 and "NFINDUP" on page 173 to do searches which do not start with *text*.
5. If SET WRAP OFF is in effect, the search continues until the top of file.

   If SET WRAP ON is in effect, the search continues through the entire file and stops when the line where it started is reached again. When a wrap occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```
6. The FINDUP subcommand updates the LASTLORC buffer. See "SET LASTLORC" on page 281.

Responses

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

Messages and Return Codes

545E Missing operand(s) [RC=5]
586E Not found [RC=2]
592W Wrapped ....

where return codes are:

0	Normal
2	No line was found
5	Missing operand(s)

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

FORWARD

```
          ┌─1─┐
►►──FOrward──┼───┼────────────────────────────────────────────────────────────►◄
          ├─n─┤
          └─*─┘
```

Purpose

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

Operands

n is the number of screen displays you want to scroll forward. If *n* is not specified, the screen scrolls forward for one display.

* moves the line pointer to the End of File line.

Usage Notes

1. The editor assigns the FORWARD subcommand to the PF8 key.
2. Issuing `FORWARD 0` from anywhere in the file makes the first line of the file the new current line.
3. 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.

Examples

In the following example, the FORWARD subcommand is being used to scroll forward 3 screens.

Current Line:

```
===== I want to scroll forward 3 screens.
fo 3
```

Your display screen will scroll forward 3 screens.

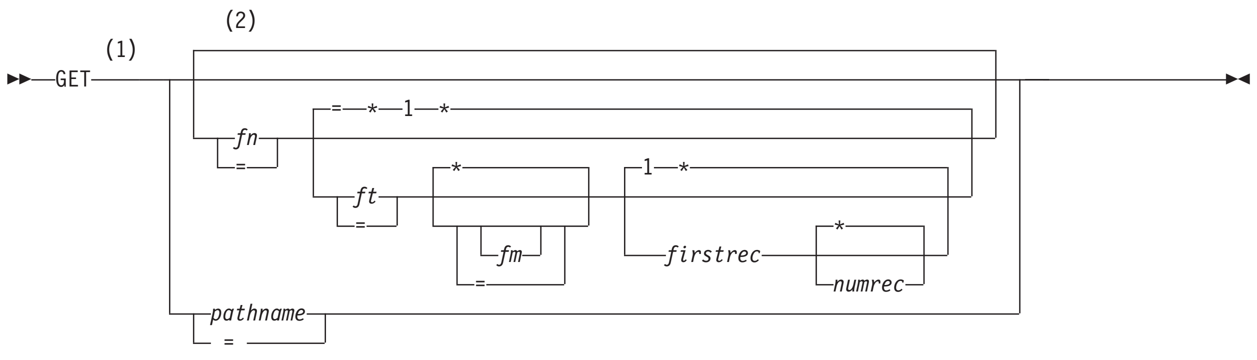
Messages and Return Codes

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

where return codes are:

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

GET



Notes:

1 The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the byte file system (BFS) format required for BFS files (*pathname*).

2 If *fn*, *ft*, and *fm* or *pathname* are not specified, lines saved by a previous PUT or PUTD subcommand in a temporary SFS or minidisk file are inserted.

Purpose

Use the GET subcommand to insert all or part of a specified CMS file after the current line of the file you are editing. Or use the GET subcommand to insert *all* of a specified byte file system (BFS) regular file after the current line of the file you are editing. You can also use the GET subcommand *without operands* to insert lines saved in a temporary SFS or minidisk file by a previous PUT or PUTD subcommand.

Operands

fn is the file name of the SFS or minidisk file that contains the data to be inserted into the file you are editing. If *ft* is not specified, the file type of the file you are editing is assumed.

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

fm is the file mode of the file that contains the data to be inserted. If *fm* is not specified, all of your accessed modes are searched for the file.

firstrec
indicates the record number of the first record you want to insert when the file is an SFS or minidisk file. 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 *firstrec* specifies. If *numrec* is not specified, or is specified as *, the remainder of the file between *firstrec* and the end of the file is inserted.

pathname
is the name of the BFS file that contains the data to be inserted into the file you

are editing. See “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the different forms of the BFS path name.

The length of the path name entered on this subcommand is limited. Use `SET PNAME` *pathname_fragment1* (or `PN` *pathname_fragment1*) and `SET PNAME APPEND` *pathname_fragment2* one or more times, and then use `GET =` if you wish to use a longer path name. Refer to Chapter 1, Chapter 1, “Rules and Conventions,” on page 1 for more information about the maximum lengths allowed for XEDIT subcommands.

Usage Notes

1. For SFS and minidisk files, the GET operand list is positional; if you omit one operand *except* for file mode, which is optional, you cannot specify any operands that follow. Therefore, if you want to specify *firstrec* and *numrec*, you must specify the file name and file type of the file.

   For BFS files, the editor will return an error if any operands follow *pathname*.
2. If you do not specify *firstrec* and *numrec*, the editor inserts the entire file.
3. If the record length of the records in the file containing the data to be inserted exceeds that of the file being edited, records are truncated and a message is displayed, unless SET SPILL ON or SET SPILL WORD is in effect. In that case, a message is displayed indicating the text has been spilled, and the characters beyond the record length are inserted in the file as one or more lines, starting with the first character or word that would have gone beyond the record length. If the record length is shorter, the records are padded with blanks to the record length of the file being edited and inserted in the file.
4. If the editor fills up available storage while executing a GET request, it may not be able to copy all of the file.
5. The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the BFS format required for BFS files (*pathname*). NAMETYPE may be set using an XEDIT option, or by the SET NAMETYPE subcommand.
6. When NAMETYPE CMS is in effect, specifying an equal sign (=) for the *fn, ft,* and *fm* operands means the corresponding FNAME, FTYPE, and FMODE values are used. If not specified as an equal sign, the operand will be translated to uppercase.

   When NAMETYPE BFS is in effect, specifying an equal sign (=) for the *pathname* operand means the value set using SET PNAME will be used. The path name is converted to uppercase only when SET CASE UPPER is in effect.

   If you change the NAMETYPE value, and then attempt to use an equal sign without setting the appropriate FNAME, FTYPE, or PNAME values, the editor returns an error. If you change the NAMETYPE value to CMS and then use an equal sign without setting FMODE, it defaults to file mode A.
7. If you issue a GET subcommand for a file in packed format, the editor does not unpack the file.
8. If operands are not specified, lines saved by a previous PUT or PUTD subcommand are inserted. The editor uses a temporary SFS or minidisk file to preserve these lines even if the file being edited is a BFS file.

Examples

This section shows examples of the GET subcommand. For more information, see *z/VM: XEDIT User’s Guide*.

Example 1

The following example shows how to insert all lines previously stored by a PUT or PUTD.

```
get
```

Example 2

The following example shows how to insert the entire file, FILE2 DATA, after the current line of this file.

```
get file2 data
```

Example 3

The following example shows how to insert the first ten lines of FILE2 DATA.

```
get file2 data 1 10
```

Example 4

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 you are editing two files.

   Move the line pointer to the beginning of lines to be inserted (use CLOCATE, UP, or NEXT, and so forth).
3. `put target`

   Stores lines from current line up to the target line.
4. `quit`

   The first file, FILE1 MINE, reappears on the screen.
5. `get` (no operands)

   The lines are inserted.

Example 5

The following example shows how to insert lines from a BFS file after the current line.

```
set nametype BFS
get /../VMBFS:filepoolid:filespaceid/pathname/filename
```

Responses

The last line inserted becomes the new current line.

When the end of the file that is being inserted is reached, the following message is displayed:

```
564W  EOF reached
```

Messages and Return Codes

002E File [*fn ft fm*] not found[: *pathname*] [RC=28]
033E File is not a regular BFS file [RC=32]
048E Invalid filemode *mode* [RC=24]
054E Incomplete [or incorrect] fileid specified [RC=24]
062E Invalid character in fileid {*fn ft fm*|*pathname*} [RC=20]
069E Filemode *mode* not accessed [RC=36]

104S	**Error *nn* reading file *fn ft fm* from disk or directory [RC=31, 55, or 100]**
156E	**Record *nnn* not found--file *fn ft fm* has only *nnn* records [RC=32]**
229E	**Unsupported OS dataset, error *nn* [RC=80, 81, 82, 83, or 84]**
512E	**This is not allowed in CMS subset mode [RC=100]**
520E	**Invalid operand: *operand* [RC=5]**
521E	**Invalid line number [RC=5]**
543E	**Invalid number: *number* [RC=5]**
554E	**Not enough virtual storage available [RC=104]**
557S	**No more storage to insert lines [RC=4]**
562E	**No line(s) saved by PUT(D) subcommand [RC=28]**
563W	**Records {truncated|spilled} [RC=3]**
564W	**EOF reached**
565W	**EOF reached; records {truncated|spilled} [RC=3]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code *rc* and reason code *rs* [RC=55, 104]**
1138E	**File sharing conflict for file {*fn ft fm*|*pathname*} [RC=70]**
1262S	**Error *nn* opening file *fn ft fm* [RC=31, 55, 99, or 100]**
2105E	**Permission is denied [24]**
2131E	**[FNAME] [FTYPE] [PNAME] {is|are} not set and NAMETYPE {CMS|BFS} is in effect [RC=24]**
2134E	**Return code *bpxrc* and reason code *bpxrs* given on call to *rtnname* [for path name *pathname*] [RC=100]**
2154E	**File {*fn ft fm*|*pathname*} is migrated and implicit RECALL is set to OFF [RC=50]**
2155E	**DFSMS/VM error occurred during creation or recall of file {*fn ft fm*|*pathname* [RC=51]**
2526E	**File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code *acs rcode* [RC=51]**

where return codes are:

0	Normal
3	Truncated or spilled
4	No more storage available to insert lines
5	Invalid operand or (line) number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
20	Character in file name, file type, or path name not valid
24	File mode or path name not valid
28	Source file not found
31	A rollback occurred
32	Record *firstrec* is beyond end of file, or BFS file is not a regular file, or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents

84	Unsupported OS data set
99	A required system resource is not available
100	Error reading file from disk, directory, or byte file system
104	Insufficient storage available

HELP (Macro)

```
        ┌─MENU─┐
►►──Help──┼──────┼──────────────────────────────────────────────────►◄
        ├─Help─┤
        ├─TASK─┤
        └─name─┘
```

Purpose

Use the HELP macro to get online information about XEDIT subcommands, macros, and messages or to invoke the HELP command. The z/VM HELP Facility allows you to display an individual XEDIT HELP file, a menu that lists all the XEDIT HELP files, or a series of menus that list XEDIT HELP files organized by various editing tasks. HELP files contain brief information, detailed descriptions, formats, parameters, options, usage notes, and related information for the XEDIT subcommands and macros.

Operands

MENU
displays a list of all XEDIT subcommand and macro HELP files, and it is the default.

Help
displays how to use the XEDIT macro HELP.

TASK
displays a list of some task-oriented HELP files.

name
can be any XEDIT subcommand name or XEDIT macro name. Specifying a name displays a description of the subcommand or macro. If name is not an XEDIT subcommand or macro, the entire HELP subcommand is transferred to the HELP command, where it must follow the CMS HELP command syntax (see *z/VM: CMS Commands and Utilities Reference*).

Usage Notes

1. XEDIT subcommand or macro abbreviations can be used in the name operand.
2. Use the HELP command format to display information on CMS as well as XEDIT error messages. For example:

   ```
   help dmsnnnnt or help dmsxxxnnnnt
   ```

   where *xxx* is the module name, *nnnn* is the message number, and *t* is the message type.
3. HELP MENU is initially set to the PF1/PF13 key.
4. Task menus are available in the RELATED sections of the SET, QUERY, and PREFIX commands. These RELATED task menus list and briefly describe the SET, QUERY, and PREFIX operands available in XEDIT.
5. Error message text information consists of a reference to the proper manual for information. Although the RELATED option is available, few examples of RELATED information files appear in the HELP files. But you can still use the

ERRORS and RELATED options to tailor your own HELP files. For information on tailoring your own HELP files, see *z/VM: CMS User's Guide*.

6. If you are viewing a command HELP file, you can issue the MOREHELP command from the command line. For example, if you are viewing the HELP file for the XEDIT CHANGE command and then decide you want to see the format section for that command, you can enter the following on the command line:

   ```
   MOREHELP (FORMAT
   ```

   The format section of the CHANGE command is then displayed.
7. If you are in a GUI environment, XEDIT HELP from the command line is displayed in a separate GUI window.

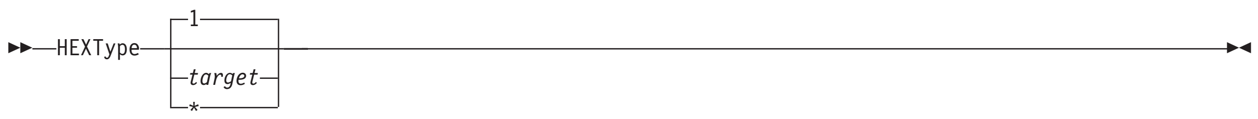
Messages and Return Codes

From the z/VM HELP Facility.

where return codes are:

0	Normal
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
any number >10	Standard CMS HELP command return codes

HEXTYPE (Macro)



Purpose

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

Operands

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 *target* is not specified, only the current line is displayed both ways.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

*
displays the rest of the file both ways (hexadecimal and EBCDIC).

Usage Notes

1. If you are in a GUI environment, messages that do not fit in the message area are displayed in the XEDIT Messages window.

Examples

In the following example, the HEXTYPE macro is used to display the current line in hexadecimal and EBCDIC.

Current Line:

```
===== To be or not to be -

hextype

E3964082 85409699 409596A3 40A39640 82854060

 T o   b e   o r   n o t   t o   b e   -
```

Responses

Each line specified is displayed first in hexadecimal and then in EBCDIC.

Only the data within the first pair of verify columns is displayed.

The line pointer moves to the last line displayed.

Messages and Return Codes

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

581E Subcommand is not valid in extended mode [RC=3]

where return codes are:

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

INPUT

```
                (1)
►►──Input──┬──────┬──────────────────────────────────────────────────────────►◄
           └─line─┘
```

Notes:

1 If *line* is not specified, you will leave edit mode and enter input mode.

Purpose

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.

Operands

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 is in effect. 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.

When you issue the INPUT subcommand without an operand, the screen display changes in the following ways:

1. The phrase `DMSXMD573I Input mode:` appears in the message area, and `Input-mode` *n* `File(s)` appears in the status area of the screen.
2. The phrase `* * * Input Zone * * *` appears in the command line.
3. The prefix area disappears from the display.
4. All file lines after the current line disappear from the display. Lines prefilled with blanks appear in their place. (You can use the SET MASK subcommand to fill this area with something other than blanks.) This blank area, between the current line and the command line, is the input zone.
5. The cursor is placed on the first line of the input zone, which is the blank line immediately after the current line. You can then type in new lines of data in the input zone.

   Only data typed in the columns the SET VERIFY subcommand defines through the truncation column (which the SET TRUNC subcommand defines) is accepted. Data entered beyond the truncation column or outside the current SET VERIFY settings is lost.

Usage Notes

1. When you fill up the screen but wish to stay in input mode and type in more lines, press ENTER once. The lines you typed move to the top half of the screen, and the last line you typed becomes the new current line; it is followed by blank lines. If AUTOSAVE is set for your editing session, you may receive the message, `DMSXMD510I AUTOSAVED as` *fn ft fm*, depending on the value set on the SET AUTOSAVE subcommand.

2. When you are finished typing in data and want to return to edit mode, press ENTER twice. The last line you typed in input mode becomes the current line, the screen layout is restored, the phrase `DMSXMD587I XEDIT:` appears in the message area, and `X E D I T` *n* `File(s)` appears in the status area of the screen. (If you did not type in new lines while in input mode, you can return to edit mode by pressing ENTER once.)
3. You can vary the size of the input zone by using the SET CURLINE subcommand to change which line on the screen is the current line. For a larger input zone, move the current line to the top of the screen; for a smaller input zone, move the current line lower on the screen.

   If you move the current line to the last available line in the file area, you cannot enter data unless you also move the command line (by using the SET CMDLINE subcommand). In general, you should leave space between the current line and the bottom of the screen for input mode.
4. 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.
5. When you use PF or PA keys in input mode, the editor automatically exits from input mode, enters edit mode to execute the subcommand associated with the PF or PA key, and returns to input mode. Therefore, you should carefully select which PF or PA keys you use in input mode. For example, if you press a PF key assigned to the FORWARD subcommand, the editor scrolls the screen forward and then returns you to input mode, but the input area will be on the next screen. In addition, some PF or PA keys are meaningless when used in input mode, for example, a PF key assigned to the ? subcommand.

   PF or PA keys particularly useful in input mode are those assigned to TABKEY, SPLTJOIN (or SPLIT and JOIN), NULLKEY, and RGTLEFT.
6. On a typewriter terminal:
   a. Pressing RETURN causes any line typed in to be inserted into the copy of the file kept in storage.
   b. If SET IMAGE ON is in effect, 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 "SET ESCAPE" on page 267) as an XEDIT subcommand. The subcommand is executed and input mode is reentered automatically. (You cannot use SET ESCAPE on a terminal in DISPLAY mode.)
   d. Enter a null line to leave input mode and return to edit mode.
7. If SET HEX ON is in effect, the line can be specified in hexadecimal. For example, INPUT X'C1C2C3'.
8. When you enter the INPUT subcommand with text specified (that is, in the form INPUT *line*) and SET SPILL OFF is in effect (the default), characters entered beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters entered beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.

   When you enter the INPUT subcommand with no text specified (that is, in the form INPUT) and input mode is entered, data is handled as described in the *line* operand description.
9. If you enter the LINEND symbol while in input mode and press ENTER, your line is not entered as separate lines in the file. Instead, it is shown as a string with the LINEND symbol appearing literally.

10. When you are at the end of file (or end of range), and you enter the INPUT subcommand, the last line of the file (or range) is displayed as the current line, even if that line is not within the defined scope.
11. If you have defined multiple logical screens with the SET SCREEN subcommand, entering input mode causes subcommands entered on other logical screens either to:
    a. Be ignored, if the screen contains a view of the same file
    b. Remain on the command line and not execute until input mode is exited, if the screen contains a view of a different file
12. If you are in a GUI environment, SET NULLS ON is in effect during INPUT mode.

Messages and Return Codes

503E	**{Truncated|Spilled} [RC=3]**
557S	**No more storage to insert lines [RC=4]**
614E	**Screen modifications lost. See SET FULLREAD ON to use PA keys safely [RC=8]**

where return codes are:

0	Normal
3	Truncated or spilled
4	No more space available to add lines
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Modifications lost because PA key pressed when message pending

JOIN (Macro)

```
         (1)
►►──Join──────┬─────────┬───┬─────────────┬──────────────────────────────────►◄
              └─ALigned─┘   ├─Column──────┤
                            ├─CURSOR──────┤
                            │   ┌───────┐ │
                            │   ▼       │ │
                            └───┬─colno────┬┴─┘
                                └─/string/─┘
```

Notes:

1 If no operand or only the ALIGNED operand is specified, the current line and the next line are joined.

Purpose

Use the JOIN macro to combine two or more lines into one replacement line.

You can use it to join two lines at the column pointer or at the cursor. You can also join two or more lines at a specified column number(s) or insert a specified character string(s) before appending the next line.

Operands

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

ALigned
removes leading blanks from the line being joined until there are the same number as there are on the line to which it is appended.

Column
joins the current line and the next line, which overlays the current line starting at the column pointer. The line pointer and the column pointer remain unchanged.

CURSOR
joins the line containing the cursor and the next line, which overlays the line starting at the cursor position. JOIN CURSOR is most useful when assigned to a PF key.

colno
specifies a column number in the current line where the next line is to be appended. Subsequent consecutive lines are appended to (and overlay the contents of) the current line for as many times as there are column numbers (*colno*) specified.

/string/
inserts the specified string (without delimiters) in the current line, starting at the first trailing blank location. Then the next line is appended after the string. This process is repeated for as many times as there are /*string*/ operands specified. Leading or trailing blanks are considered part of *string*.

Usage Notes

1. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. For the JOIN macro, SET SPILL OFF (the default) has the same effect as SET SPILL ON. JOIN does not truncate data.
2. The original lines that are appended as a result of a JOIN macro are deleted, unless data precedes zone 1 (see “SET ZONE” on page 371). In this case only the data following zone 1 is deleted.
3. Before using JOIN COLUMN, check to see if the column pointer is in the desired location, because the next line is appended starting at the column pointer. Use the CLOCATE subcommand to move the column pointer, if necessary.
4. The line pointer and column pointer remain unchanged.
5. JOIN is the converse of SPLIT. See also SPLTJOIN, which combines SPLIT and JOIN.
6. The cursor or the column number or string specified must fall within the current zones.

Examples

This section shows examples of the JOIN macro.

Example 1

The following example assigns PF11 to JOIN CURSOR.

```
set pf11 join cursor
```

```
===== This line is _
===== too short.
```

Note position of the cursor (_) in the first line. Pressing PF11 produces the following line:

```
===== This line is too short.
```

Example 2

The following example joins the current line and the next two, separated by semi-colons.

Current Line

```
===== .sp
===== .in 5
===== .of 3

join /;/ /;/

===== sp;.in 5;.of 3
```

Example 3

The following example uses JOIN ALIGNED to remove up to the same number of leading blanks from the line being joined to the current line.

```
=====      These lines have _
=====      leading blanks.

join al cursor

=====      These lines have leading blanks.
```

Example 4

The following example uses JOIN to join lines separated by blanks.

```
===== Electric eels
===== can discharge bursts
===== of 625 volts.

join / / / /

===== Electric eels can discharge bursts of 625 volts.
```

Messages and Return Codes

503E **{Truncated|Spilled} [RC=3]**
520E **Invalid operand:** *operand* **[RC=5]**
526E **Option** *option* **valid in display mode only [RC=3]**
561E **Cursor is not on a valid data field [RC=1]**
564W **EOF reached [RC=1]**
575E **Invalid [argument or]{JOIN|SPLIT|TABS|VERIFY|ZONE} column(s) defined [RC=5]**
585E **No line(s) changed [RC=1]**
685E **Joined lines(s) exceed zone settings [RC=5]**

where return codes are:

0	Normal
1	No line(s) changed, or cursor is not on a valid data field
3	Spilled, or operand is valid only for display terminal
5	Invalid operand, or joined line(s) exceed zone settings
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

LEFT

```
          ┌─1─┐
►►──LEft──┼───┼──────────────────────────────────────────────────────────────►◄
          └─n─┘
```

Purpose

Use the LEFT subcommand to view columns of data not currently visible on the screen. The LEFT subcommand allows you to see data *to the left of the first column* on the screen. The data moves to the right, thus allowing you to see a specified number of positions to the left of the first column.

Operands

n specifies the number of positions to the left of the first column on the screen you want to see. If *n* is not specified, one position to the left of the first column becomes visible. The total number of columns specified cannot exceed the logical record length.

Usage Notes

1. The LEFT subcommand does not cause data to be lost nor does it move the line or column pointer.
2. To get the data back to its original position, use the RIGHT *n* subcommand. See "RGTLEFT (Macro)" on page 219.
3. LEFT subcommands are cumulative. For example entering the subcommands:

   ```
   left    10
   left    10
   ```

   is equivalent to entering:

   ```
   left    20
   ```

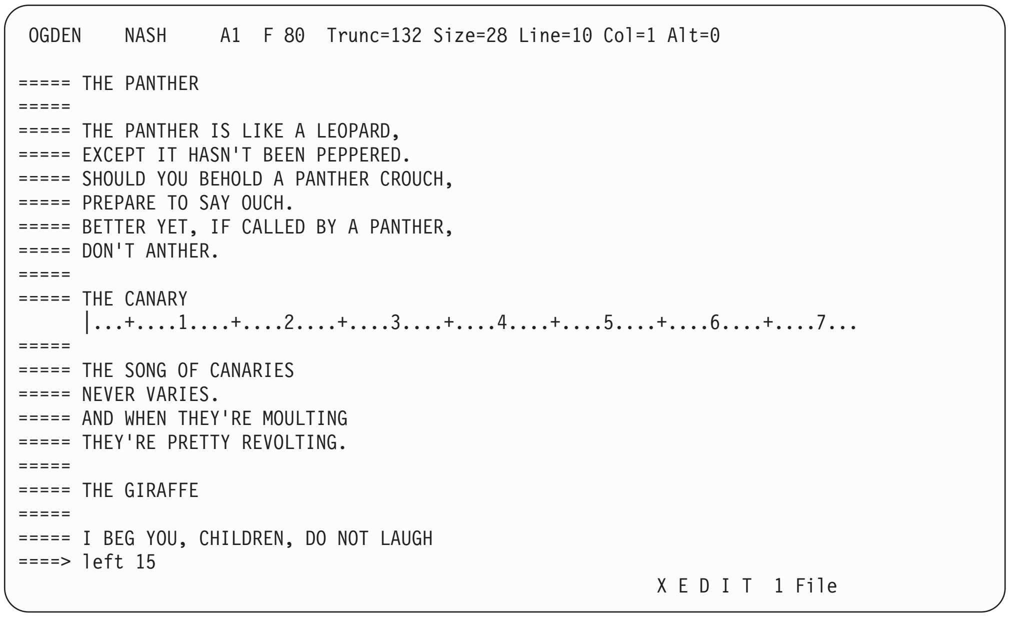
   Therefore, if several LEFT subcommands have been issued, column one on the screen might not contain the first character in a line.
4. If you have issued several LEFT or RIGHT subcommands and have forgotten the value of *n*:
   a. LEFT 0 or RIGHT 0 restores the screen to its original display.
   b. Using the SET VERIFY subcommand resets LEFT (or RIGHT) to zero.
   c. QUERY VERSHIFT displays *n* or –*n*, which is the relative position of the screen over the file, as a result of all LEFT or RIGHT subcommands.

Notes for Macro Writers

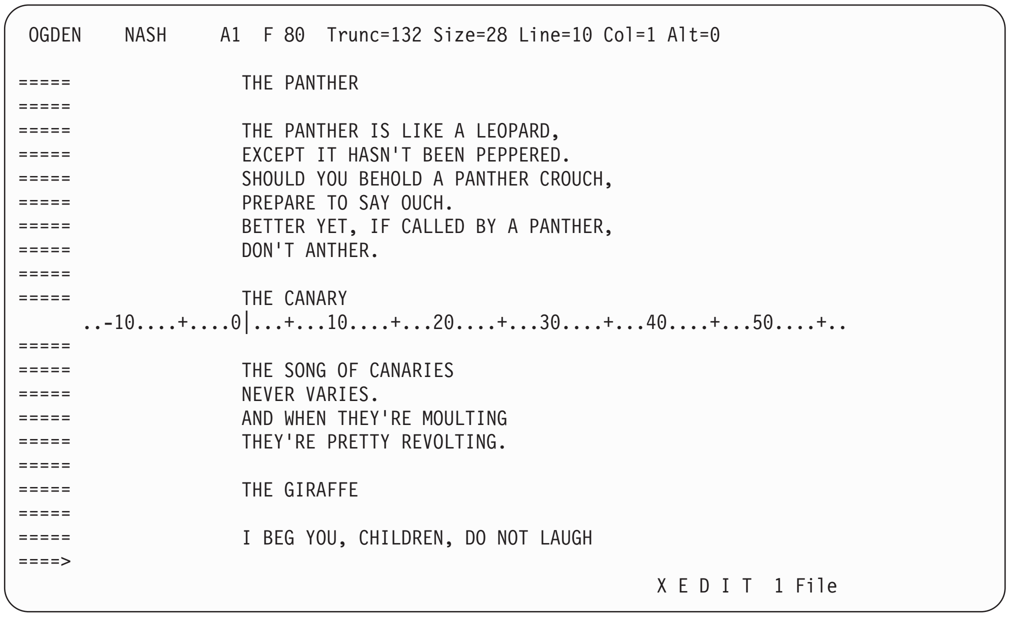
1. EXTRACT /VERSHIFT/ returns the value of *n* or –*n*.

Examples

Figure 8 on page 139 is a before-and-after example of the LEFT subcommand.



```
 OGDEN    NASH     A1  F 80  Trunc=132 Size=28 Line=10 Col=1 Alt=0

===== THE PANTHER
=====
===== THE PANTHER IS LIKE A LEOPARD,
===== EXCEPT IT HASN'T BEEN PEPPERED.
===== SHOULD YOU BEHOLD A PANTHER CROUCH,
===== PREPARE TO SAY OUCH.
===== BETTER YET, IF CALLED BY A PANTHER,
===== DON'T ANTHER.
=====
===== THE CANARY
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
===== THE SONG OF CANARIES
===== NEVER VARIES.
===== AND WHEN THEY'RE MOULTING
===== THEY'RE PRETTY REVOLTING.
=====
===== THE GIRAFFE
=====
===== I BEG YOU, CHILDREN, DO NOT LAUGH
====> left 15
                                                       X E D I T  1 File
```



```
 OGDEN    NASH     A1  F 80  Trunc=132 Size=28 Line=10 Col=1 Alt=0

=====               THE PANTHER
=====
=====               THE PANTHER IS LIKE A LEOPARD,
=====               EXCEPT IT HASN'T BEEN PEPPERED.
=====               SHOULD YOU BEHOLD A PANTHER CROUCH,
=====               PREPARE TO SAY OUCH.
=====               BETTER YET, IF CALLED BY A PANTHER,
=====               DON'T ANTHER.
=====
=====               THE CANARY
      ..-10....+....0|...+...10....+...20....+...30....+...40....+...50....+..
=====
=====               THE SONG OF CANARIES
=====               NEVER VARIES.
=====               AND WHEN THEY'RE MOULTING
=====               THEY'RE PRETTY REVOLTING.
=====
=====               THE GIRAFFE
=====
=====               I BEG YOU, CHILDREN, DO NOT LAUGH
====>
                                                       X E D I T  1 File
```

Figure 8. LEFT Subcommand — Before and After

Responses

The screen moves to the left relative to the file data.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
576E {Total verify width exceeds screen size (*nn***)|Total offset exceeds LRECL (***nn***)} [RC=5]**

where return codes are:

0	Normal
5	Invalid operand or number, or total verify width exceeds screen size

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

LOAD

```
                                          (2)
               (1)               ┌─*──┐  ┌─(──────────────────────────────────────────────┐
►►──LOAD──────────┬─fn──ft─────┤    ├──┼────────────────────────────────────────────────┼──────►◄
                  │             └─fm─┘  │  (3)                                           │
                  └─pathname────────────┘  └─(──────┤ Options ├──┤ Update Mode Options ├──┬────┬┘
                                                                                         └─)──┘
```

Options:

```
   ┌─WINdow──XEDIT─┐
├──┼───────────────┼──┬──────────────┬──┬──────────┬──┬──────────┬──┬───────┬──────────────────►
   └─WINdow──wname─┘  └─Width──width─┘  └─NOSCreen─┘  └─NOCLear──┘  └─NOMsg─┘

                             ┌─LOCk───┐  ┌─NOUpdate─┐  ┌─BFSLine──NL─────────────────┐
►──┬────────────────────────┬─┼────────┼──┼──────────┼──┼─────────────────────────────┼───────►
   └─MEMber──membername─────┘ └─NOLOCk─┘  └─Update───┘  └─BFSLine──┬─lrecl────────┬─┘
                                                                   ├─CRLF─────────┤
                                                                   ├─CRNL─────────┤
                                                                   ├─/string/─────┤
                                                                   └─/hexstring/──┘

   ┌─NAMetype──CMS─┐
►──┼───────────────┼──────────────────────────────────────────────────────────────────────┤
   └─NAMetype──BFS─┘
```

Update Mode Options:

```
   ┌─Seq8───┐                                                 ┌─Incr──1────┐
├──┼────────┼──┬──────────┬──┬───────┬──┬────────────┬──┼────────────┼──┬────────────────┬──┤
   └─NOSeq8─┘  ├─Ctl──fn1─┤  └─Merge─┘  └─UNtil──ft1─┘  └─Incr──incr─┘  └─SIDcode──string─┘
               └─NOCtl────┘
```

Notes:

1 If you entered the *fn ft* or the *pathname* with the XEDIT command or subcommand, it is not necessary to enter them with the LOAD subcommand.

2 The default options are shown above the main line in the options groups.

3 You can enter options in any order between the parentheses. If a default is not shown for an option, refer to the option description for the information.

For a description of the editing options, see Chapter 2, "The XEDIT Command," on page 11. Remember the options specified with the XEDIT command (or subcommand) override those specified with the LOAD subcommand.

Purpose

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 except for PROFILE and

NOPROF, which are ignored; however, the options specified in the XEDIT command override those specified in the LOAD subcommand.

Usage Notes

1. WINDOW *wname* is the name of the virtual screen and window XEDIT uses to display the file or files being edited. For more information on virtual screens and windows, see Appendix G, "XEDIT Virtual Screens and Windows" on page 487.
2. The WIDTH option specifies the amount of virtual storage to be used for one file line.

   If, upon editing an existing file, the value of WIDTH specified in the LOAD subcommand is too small to contain a file line, it is overridden by:

   a. The WIDTH value, if specified, in the XEDIT command
   b. The logical record length of the file being read.

   Therefore, the value of WIDTH specified in the LOAD subcommand cannot cause unwanted truncation.

   For newly created files, WIDTH is the value specified on the LOAD subcommand unless the value specified on the XEDIT command overrides it.

   If WIDTH is not specified in either the LOAD subcommand or the XEDIT command, its value for SFS and minidisk files is the greater of:

   a. The logical record length (LRECL) of the file, or
   b. The default logical record length associated with the file type. See Appendix A, "File Type Defaults" on page 453.

   Note that WIDTH is the only option that has a different meaning in the LOAD subcommand and XEDIT command.

   For BFS files, if WIDTH is not specified in either the XEDIT command or the LOAD subcommand, the default depends upon the BFSLINE value.

   - For BFSLINE *lrecl*, the default is *lrecl*.
   - Otherwise, the default value is 80 or the length of the longest line, whichever is longer.
3. The XEDIT variables LRECL, TRUNC, ZONE, and VERIFY are assigned default values when the LOAD subcommand is executed. During editing, the SET subcommand can change these variables:

   LRECL

   - **For SFS or minidisk files**, LRECL is the logical record length used when the file is written to disk or directory. For files with a variable (V) record format, the LRECL is assigned the value specified in WIDTH. Thus, the longest record cannot exceed the value of WIDTH.

     For files with a fixed (F) record format, the LRECL assigned is one of the following:

     – If the file existed and has been read from a disk or a directory, the LRECL assigned is the logical record length of the file.

     – If the file is new, LRECL is either WIDTH or the default logical record length assigned to the file type, whichever is smaller.
   - **For BFS files**, LRECL is assigned depending upon BFSLINE.

     – For BFSLINE *lrecl*, LRECL is assigned to *lrecl*.

     – For any other BFSLINE value, LRECL is assigned to whichever is smaller, the WIDTH or the longest record read.

TRUNC
is assigned the value of LRECL, except for fixed (F) format SFS or minidisk files, which have a default value associated with the file type.

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

VERIFY
is assigned the value of TRUNC unless otherwise specified. See Appendix A, “File Type Defaults” on page 453.

For a list of file type defaults for SFS or minidisk files, see Appendix A, “File Type Defaults” on page 453.

4. Within the profile macro, the LOAD subcommand must be the first XEDIT subcommand. If it is not, the editor automatically issues a LOAD subcommand; its operands are the same as those in the XEDIT command. (EXEC 2 statements, REXX instructions, and CMS commands can be issued before the LOAD. When a CMS command is issued before the LOAD, the command must be specifically addressed to CMS. In REXX, use ADDRESS CMS and in EXEC 2 use &COMMAND.)

   A synonym cannot be used to specify the LOAD subcommand. Multiple subcommands may not be specified with the linend character following the LOAD subcommand.
5. The profile macro can be used to prompt the user for XEDIT command options or to assign values to editing variables before issuing the LOAD subcommand. For example, a SCRIPT user might program his or her profile to use a LOAD subcommand that does defaulting of file type. For example:

   ```
   parse arg fn ft '(' options
   if ft = '' then ft = 'SCRIPT'
   LOAD fn ft '(' options
   ```

   The user might also program their profile to use a LOAD subcommand that defaults NAMETYPE to BFS. A BFS path name may be up to 1023 bytes long, is case sensitive, and may contain blanks. The untokenized, mixed case file ID is passed as a second argument string to a PROFILE written in REXX. So the user who is always editing BFS files may program their profile to use the second argument string on the LOAD command. For example:

   ```
   parse arg fn  '(' options , pathname
   options = options ' NAMETYPE BFS'
   'LOAD' pathname '(' options
   ```
6. The options specified in the LOAD subcommand have a *lower* priority than those in the XEDIT command. For example, NOUPDATE specified in the XEDIT command would override UPDATE specified in the LOAD subcommand.

   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 options specified in the XEDIT command can override. For example, if you enter:

   ```
   XEDIT fn1 ft1 fm1 (profile tst nametype cms
   ```

   Where “tst xedit” contains:

   ```
   load fn2 (nametype bfs
   ```

   The file that would be edited would be *fn2 ft1 fm1* (with data from *fn2 ft1 fm1* if it was found). Because the LOAD specified *fn2*, but not *ft2* and *fm2*, the *ft1* and *fm1* were used by default. The option NAMETYPE CMS overrides the NAMETYPE BFS and XEDIT attempts to edit a CMS file.

7. A return code of 0 or 3 from the LOAD subcommand means a successful load has been issued. If LOAD completes with any other return code, all subsequent subcommands in the profile are rejected with a unique return code 6, and the editor automatically issues a quit subcommand.
8. The WINDOW, NOCLEAR, and NOSCREEN options are ignored if you are in a GUI environment.

Responses

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

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

The editing session continues but the file is not locked.

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

```
178I  Updating fn ft fm
      Applying fn ft fm
1229I fn ft fm is empty
          .
          .
          .
180W  Missing PTF file fn ft fm
```

If the XEDIT work file, XEDTEMP CMSUT1, exists on a file mode accessed R/W as a result of a previous edit session that ended abnormally, you will receive the message:

```
024E   File XEDTEMP CMSUT1 fm already exists
```

You can use the CMS TYPE command to examine the existing file. If you decide you want to keep it, use the CMS RENAME command to give it a new file ID. If the file is incorrect or incomplete, erase it and reissue the command.

Messages and Return Codes

002E File *fn ft fm* **not found [RC=28]**
003E Invalid option: *option* **[RC=24]**
024E File XEDTEMP CMSUT1 *fm* **already exists [RC=28]**
029E Invalid parameter *parameter* **in the option** *option* **field [RC=24]**
033E File is not a regular BFS file [RC=32]
048E Invalid filemode *mode* **[RC=24]**
054E Incomplete [or incorrect] fileid specified [RC=24]
062E Invalid character in fileid {*fn ft fm***|***pathname***} [RC=20]**
065E *Option* **option specified twice [RC=24]**
066E *Option1* **and** *option2* **are conflicting options [RC=24]**
069E Filemode *mode* **not accessed [RC=36]**
070E Invalid parameter *parameter* **[RC=24]**
104S Error *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, or 100]**
109S Virtual storage capacity exceeded [RC=104]
132S File [*fn ft fm***] too large[:** *pathname***] [RC=88]**
137S Error *nn* **on STATE for** *fn ft fm* **[RC=88]**
229E Unsupported OS dataset, error *nn* **[RC=80, 81, 82, or 83]**
500E Unable to unpack file *fn ft fm* **[RC=88]**
508E LOAD must be the first subcommand in the profile [RC=3]
512E This is not allowed in CMS subset mode [RC=100]
554E Not enough virtual storage available [RC=104]

555E File {*fn ft fm*|*pathname*} already in storage [RC=4]
556I Editing existing empty file:
571I Creating new file:
622E Insufficient free storage (for {MSGLINE|PFkey/PAkey|synonyms})
915E Maximum number of windows already defined [RC=13]
927E The virtual screen must contain at least 5 lines and 20 columns [RC=24]
928E Command is not valid for virtual screen *CMS* **[RC=12]**
1019E Network File System name is not allowed [RC=32]
1020E Foreign host cannot be reached. The request returned return code *rc* **and reason code** *rs* **[RC=55, 104]**
1138E File sharing conflict for file {*fn ft fm***|***pathname* **[RC=70]**
1214W
File *fn ft fm* **already locked SHARE**
1215E File *fn ft fm* **is locked or in use by another user [RC=70]**
1229E *fn ft fm* **is empty [RC=88]**
1229I *fn ft fm* **is empty**
1262S Error *nn* **opening file** *fn ft fm* **[RC=31, 55, 70, 76, 99, or 100]**
1299W
Warning: Not authorized to lock file *fn ft fm*
1300E Error *nn* **{locking|unlocking} file** *fn ft {fm|dirname}* **[RC=55, 70, 76, 99, or 100]**
2105E Permission is denied [24]
2134E Return code *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname***] [RC=100]**
2154E File {*fn ft fm***|***pathname***} is migrated and implicit RECALL is set to OFF [RC=50]**
2155E DFSMS/VM error occurred during creation or recall of file {*fn ft fm***|***pathname***} [RC=51]**
2526E File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code *acs rcode* **[RC=51]**

Messages with Member Options

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

Messages with Update Options

007E File *fn ft fm* **is not fixed, 80-character records [RC=32]**
007E File *fn ft fm* **does not have a logical record length greater than or equal to 80 [RC=32]**
007E File *fn ft fm* **does not have the same format and record length as** *fn ft fm* **[RC=32]**
007E File *fn ft fm* **is not fixed record format [RC=32]**
104S Error *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 32, or 55]**
174W Sequence error introduced in output file: *seqno1* **to** *seqno2* **[RC=32]**
178I Applying *fn ft fm*
179E Missing or invalid MACS card in control file *fn ft fm*
180W Missing PTF file *fn ft fm*
183E Invalid {CONTROL|AUX} file control card [RC=32]
184W ./ S not first card in update file--ignored [RC=32]
185W Non numeric character in sequence field *seqno* **[RC=32]**
186W Sequence number [*seqno1***] not found [RC=32]**
207W Invalid update file control card [RC=32]

210W	**Input file sequence error:** *seqno1* **to** *seqno2* **[RC=32]**
317E	**Number of AUX file types in control file** *fn ft fm* **exceeds 32 [RC=32]**
570W	**Update** *ft* **specified in the UNTIL option field not found**
597E	**Unable to merge updates containing ./ S cards [RC=32]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=31, 55, 70, 76, 99, or 100]**

where return codes are:

0	Normal
3	LOAD has already been issued
4	File is already in storage
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Command is not valid for virtual screen
13	Maximum number of windows already defined
20	Character in file name, file type, or path name not valid
24	Invalid parameters or options
28	Source file not found (UPDATE MODE), or library not found (MEMBER option), or specified PROFILE macro does not exist, or file XEDTEMP CMSUT1 already exists
31	A rollback occurred
32	Error during updating process, or file is not a library, or library has no entries, or file is not fixed, 80-character records, BFS file is not a regular file, or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
88	File is too large and does not fit into storage, a previous MACLIB function was not finished, or an unsupported function was attempted with an empty file
99	A required system resource is not available
100	Error reading the file into storage
104	Insufficient storage available

LOCATE

```
►►──┬────────┬──target──┬────────────┬──────────────────────────────────►◄
    └─Locate─┘          └─subcommand─┘
```

Purpose

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 after the current line. Optionally, you can specify an XEDIT subcommand; it is executed starting at the specified target.

Note: To display **all** lines that correspond to the specified string, use the ALL macro.

Operands

Locate
is the subcommand name but is optional. The target operand itself implies the LOCATE subcommand.

target
defines the line that is to become the new current line. It can be specified as an absolute line number, a relative displacement from the current line, a line name, a simple string expression, or a complex string expression. A complete description of targets follows.

subcommand
is any XEDIT subcommand, which is executed starting at the specified target.

Usage Notes

The ability to locate a line with a target is one of the editor's most versatile functions. A target is not only the operand of the LOCATE subcommand but is also 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" on page 317) and between the beginning and end of each line (or between the left and right zones — see "SET ZONE" on page 371).

You can specify a target in the following ways:

1. An *absolute line number* is a colon (:) followed by a file line number or an asterisk. An asterisk indicates the null End of File (or End of Range) line. For example:

   ```
   :8
   ```

   makes file line number 8 the new current line.

   You can also search for a character string by preceding it with a colon. For example:

   ```
   :ABC
   ```

   searches for the first occurrence of ABC. If the character string starts with a number, an error will occur.

2. A *relative displacement* from the current line is an integer that can 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.

   You can also specify a relative displacement as an asterisk (*), which means the Top of File (–*) or the End of File (+* or *) line. When you specify an asterisk 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`

   copies lines 5, 6, and 7 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 1 - 8 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:

   ```
   .PART
   ```

   locates the file line whose name is PART and makes it the current line.

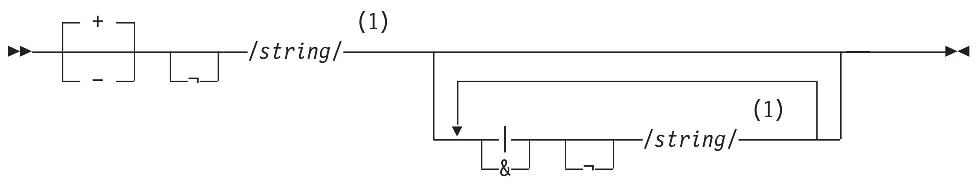
   The SET CASE (UPPERCASE or MIXED) option determines whether the line name is translated to uppercase. Additionally, in locating a line name target, uppercase and lowercase characters are significant regardless of the SET CASE option RESPECT or IGNORE.

4. A *string expression* defines a group of characters to be located. The characters in the string must be delimited by any character that does not appear in the string itself. However, if you use the XEDIT special characters (+ – .) 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 examples that follow, a diagonal (/) is used.

   If SET HEX ON is in effect, you can specify a string in hexadecimal notation, and the editor searches for its EBCDIC equivalent. For example, if you specify a string as /X'C3D4E2'/, the editor searches for the string CMS.

   If SET ESCAPE ON is in effect, the commonly used LOCATE delimiter (/) may be interpreted as the ESCAPE character. To avoid this problem, use the LOCATE subcommand explicitly (for example, LOCATE /*string*/).

   The general format for a string expression is:



Notes:

1 The final delimiter (/) is optional after the last *string*. Leading or trailing blanks are considered part of *string*.

+ or –
The search direction is toward the end of the file (+) or toward the top of the file (–). If the sign is omitted, the default is plus (+).

¬ *NOT* symbol (Locate something that is not the specified *string*.)

string
Character (or hexadecimal) string. The trailing delimiter may be necessary in certain circumstances. For example, if the first *string* has trailing blanks, use the trailing delimiter to indicate where the string ends.

& *AND* symbol (ampersand) (Locate the line containing all of the *string*s separated by an &.)

| *OR* symbol (vertical bar) (Locate the line containing any of the *string*s, separated by OR symbols, starting with the first *string* specified.)

For example:

`/horse/`
searches downward in the file, beginning after the current line, for the first line that contains *horse* and makes it the current line.

`¬/house/`
searches downward in the file for the first line that does *not* contain *house* and makes it the current line.

`/horse/ & /house/ | /hay/`
searches downward in the file for a line that contains either both *horse and house or* a line that contains *hay*, whichever comes first.

Targets *anded* together can overlap. For example, L/This/&/history/ could find the string “Thistory” as well as the string “This history”.

`/horse/|¬/house/`
searches downward in the file for the first line that contains *horse or* does not contain *house*.

`-/X'C1'/|/X'C2'/`
searches upward for the first line containing *either or both* of the strings specified here in hexadecimal.

If SET HEX ON is in effect, the editor locates a line containing A or B. If SET HEX OFF is in effect, the editor locates a line containing X'C1' or X'C2'.

`//`
advances the line pointer by one line.

5. A *complex string expression* has the same format as a simple string expression. A complex string is 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 the difference between uppercase and lowercase is to be significant in locating a string target.

`SET ETARBCH`

See Appendix F, "Using Double-Byte Character Sets," on page 467.

`SET SPAN`

allows you to specify if a string target must be included in one file line or if it can span a specified number of lines.

`SET VARBLANK`

allows you to control whether or not the number of blank characters between two words is significant in a target search.

For example:

LOCATE and SET ARBCHAR

```
set arbchar on .
/air.plane/
```

would locate in the text either one of the following:

```
the airplane was landing
cold air surrounded the plane
```

LOCATE and SET CASE

```
set case m ignore
/computer/
```

would locate in the text any of the following:

```
computer
Computer
comPUTer
```

LOCATE and SET SPAN

If SET SPAN OFF is in effect, a string must be contained within one file line in order to match a target.

If SET SPAN ON is in effect, a string may start on one line and continue on *n* lines (as specified in the SET SPAN subcommand) and still match a target. In this case, the line containing the beginning of the string becomes the current line.

LOCATE and SET VARBLANK

```
set varblank on
/the  house/
```

will locate in the text either of the following:

```
the house
the                house
```

Note: Target as discussed here is a *line target*, not to be confused with a *column-target*, which is the operand of only the CLOCATE and CDELETE subcommands.

Usage Notes for LOCATE Targets

1. LOCATE targets and SET STAY: If SET STAY OFF is in effect and the target is not located, the new current line is the bottom (or top) of the file (or range).

   If SET STAY ON is in effect and the target is not located, the line pointer remains unchanged.

Note that SET STAY only applies when the target is not located and the search is issued to the end of file or end of range (that is, SET WRAP OFF is in effect).

2. LOCATE targets and SET WRAP: If SET WRAP OFF is in effect, the search for a string expression target stops with the end of file or range (or top of file or range).

   If SET WRAP ON is in effect, the editor wraps around the file and continues the search for a string expression up to and including the line preceding the current line. If a range has been defined, the editor wraps around the bottom of the range and continues at the top of the range (or if the search is in the other direction, the editor wraps around the top and continues at the bottom).

   When a wrap occurs under these conditions, the following warning message is displayed:

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

   Note that if SET WRAP ON is in effect and the target is not located, the line pointer remains unchanged regardless of the SET STAY setting.
3. The LOCATE subcommand updates the LASTLORC buffer. See the SET LASTLORC subcommand.
4. For more information on the LOCATE subcommand, see Appendix B, “Effects of Selective Line Editing Subcommands,” on page 455.
5. If the LOCATE subcommand reaches its target successfully (either return code 0 or 1), *subcommand* is executed.

Notes for Macro Writers

1. In a macro, an implicit backward locate (for example, –3) is interpreted by EXEC 2 as a label. To avoid this problem, use the LOCATE subcommand explicitly (for example, LOCATE –3), or use the COMMAND subcommand (COMMAND –3).

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]
546E Target not found [RC=2]
592W Wrapped ....

where return codes are:

0	Normal
1	TOF or EOF reached
2	No target line was found
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
nn	Return code from subcommand following LOCATE

LOWERCAS

```
           ┌─1──────┐
►►──LOWercas──┼────────┼──────────────────────────────────────────────►◄
           ├─target─┤
           └─*──────┘
```

Purpose

Use the LOWERCAS subcommand to change all uppercase letters to lowercase in one or more lines of the file, starting with the current line.

Operands

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 omit *target*, only the current line is translated.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

* the rest of the file is translated.

Usage Notes

1. The LOWERCAS subcommand does not alter the setting of the SET CASE subcommand.

Examples

In the following example, LOWERCAS translates the current line to lowercase.

Current Line:

```
===== Tortoises of the Galapagos Islands can live to be 100 years old.

low

===== tortoises of the galapagos islands can live to be 100 years old.
```

Responses

1. All uppercase letters within the current zones are changed to lowercase.
2. If multiple line targets are specified, and the subcommand is executed, the current line pointer:
   a. Is unchanged, if SET STAY ON is in effect
   b. Moves to the last line translated, if SET STAY OFF is in effect (the default).

Messages and Return Codes

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

where return codes are:

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

LPREFIX

```
                (1)
               ┌────┐
►►──LPrefix────┴text┴──────────────────────────────────────────────►◄
```

Notes:

1 If *text* is not specified, the pending list is executed immediately.

Purpose

Use the LPREFIX subcommand to simulate writing in the prefix area of the current line. LPREFIX can be used on typewriter terminals to utilize some of the features of prefix subcommands and macros, as well as on display terminals, regardless of the appearance or position of actual prefix areas on the screen (see "SET PREFIX" on page 315).

Operands

text
: specifies up to five characters. All prefix subcommands and macros are put in a *pending list* before they are called. If *text* is not specified, the pending list is executed immediately. If text is specified, the text is placed in the pending list for the current line and the pending list is then executed. For more information about the pending list, see "SET PENDING" on page 304.

Usage Notes

1. Note, the action taken after LPREFIX is the same as the action taken when you enter the same *text* subcommand directly in the prefix area of the current line and press ENTER.
2. LPREFIX *text* is equivalent to issuing SET PENDING ON *string* and pressing ENTER again.
3. LPREFIX cannot be issued from a prefix macro.
4. The prefix subcommands and macros useful on a typewriter terminal through LPREFIX are:

```
D,DD
","" 
C,CC
M,MM
X,XX
<,<<
>,>>
.xxxx
P
F
A
I
```

Examples

In the following example, the LPREFIX subcommand uses some of the prefix subcommands.

Nefarious Nelly put her grocery list online. Every day she added more items. At the end of the week, she wanted to organize the list in the following order: meat, fruit, vegetables. Working at a typewriter terminal, she used the LPREFIX subcommand to move the grocery items appropriately. The following is a listing of her file's contents:

```
TOF:
Toad Toes
Newt Eyes
Salmon Scales
Root of Hemlock
Baneberries
Poison Plums
Paltry Peaches
EOF:
```

Nelly moves to the top of the file by typing TOP and then moves down to the line BANEBERRIES by typing N5.

```
top
```

```
TOF:
```

```
n5
```

```
Baneberries
```

To move the last three lines of the file (the fruit) so they follow Salmon Scales (the last line of the meat group), Nelly used the LPREFIX subcommand as shown in the following sequence:

```
lprefix mm

n2
```

```
Paltry Peaches
```

```
lprefix mm

top
```

```
TOF:
```

```
n3
```

```

Salmon Scales
```

```
lprefix f
```

The resulting file:

```
top
```

```

TOF:
```

```
t*
```

```

TOF:
Toad Toes
Newt Eyes
Salmon Scales
Baneberries
Poison Plums
Paltry Peaches
Root of Hemlock
EOF:
```

Responses

Any response from the executed prefix subcommand or macro is displayed.

Messages and Return Codes

509E *subcommand* **subcommand not valid from a prefix macro [RC=4]**
520E Invalid operand: *operand* **[RC=5]**

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

where return codes are:

nn	Return code of the prefix subcommand or macro specified as operand
0	Normal
4	Invalid when issued from a prefix macro
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

MACRO

```
►►──MACRO──┬───────┬──────────────────────────────────────────────────►◄
           └─macro─┘
```

Purpose

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

Operands

macro
is an XEDIT macro name and its arguments (if any).

The first word in the *macro* is assumed to be an XEDIT macro name. It starts with the first nonblank character after the subcommand name MACRO and ends with the first character followed by a blank. The name can be 1 - 8 characters; 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 nonalphabetic characters. It does not follow the usual parsing rule of separating alphabetic characters from immediately following nonalphabetic characters. For example, N2 usually means NEXT 2. MACRO N2 means *execute the macro named N2.*
3. The macro may not reside in the byte file system (BFS).

Examples

The following examples shows how a macro with an argument can be invoked.

Example 1

This example invokes the macro file N XEDIT with the argument ABC.

```
macro N ABC
```

Example 2

This example invokes the macro file N2 XEDIT with the argument ABC.

```
macro N2 ABC
```

Responses

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

Messages and Return Codes

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

where return codes are:

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

MERGE

```
                                  ┌─1─────┐
►►──MErge──target1──target2───────┼───────┼──────────────────────────────────────────►◄
                                  └─colno─┘
```

Purpose

Use the MERGE subcommand to combine two sets of lines. The first set of lines is deleted and the second set is modified in place.

Operands

target1
: defines the number of lines to be merged, starting with the current line up to, but not including, *target1*. *target1* defines the first group of lines you wish to merge with a second group of lines.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

target2
: defines the beginning of the second group of lines to be merged with the first.

colno
: specifies the column number to which column 1 of the first group of lines being merged is to be shifted. The first group is shifted to the right of the second group and then overlays the second group. This allows the merger of two columns of data, side by side. The default *colno* is 1.

Usage Notes

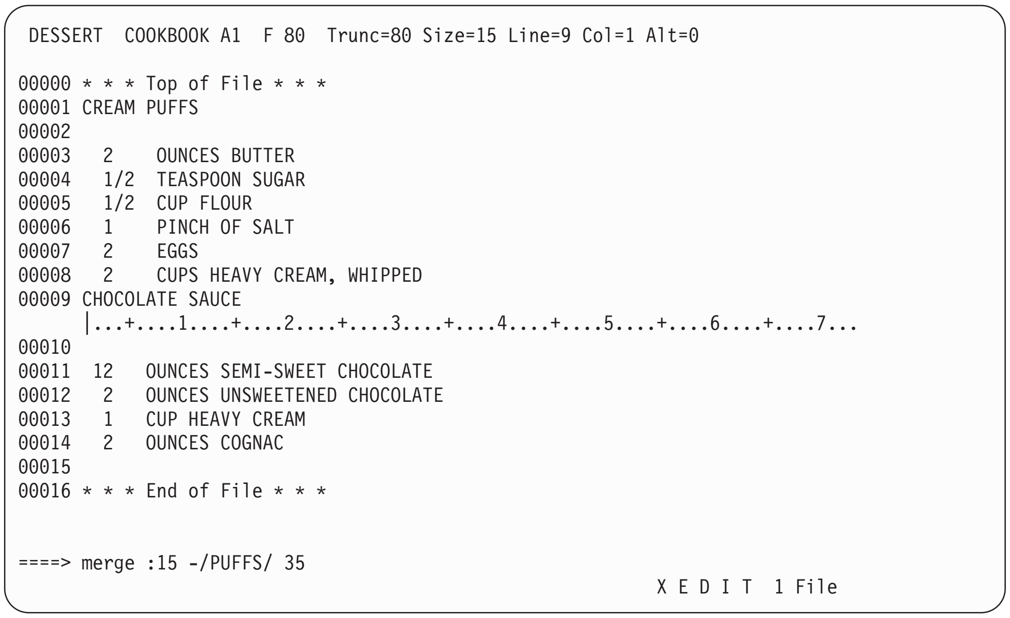
1. MERGE shifts the first group of lines and then overlays the second group of lines. It is similar in function to the OVERLAY subcommand, except that MERGE does not give special treatment to underscore characters.
2. Following the MERGE with the second set of lines, the first set of lines is deleted. The lines can be recovered by using the RECOVER subcommand.
3. The first group of lines cannot overlap the second group of lines.
4. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column.
5. When combining two lines, the merge handles blanks as follows:

   ```
   First line    Second line    Result line
     blank         blank          blank
       x           blank            x
     blank           y              y
       x             y              x
   ```

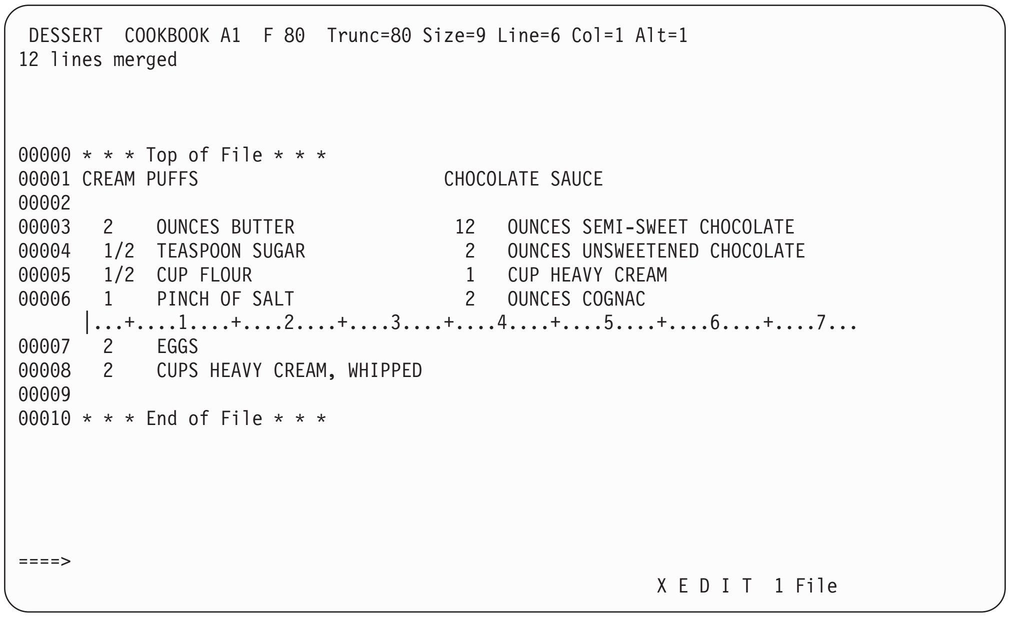
   where 'First line' designates any of the lines in the group starting at the current line, and 'Second line' designates any of the lines in the group starting at the second target parameter. The blank columns of the line to be modified are replaced with the corresponding columns from the line in the first merge group; however, blanks in the first merge group do not replace character data in the line to be modified.

Examples

Figure 9 is a before-and-after example of the MERGE subcommand. Line 9 is the current line.



```
 DESSERT  COOKBOOK A1  F 80  Trunc=80 Size=15 Line=9 Col=1 Alt=0

00000 * * * Top of File * * *
00001 CREAM PUFFS
00002
00003   2    OUNCES BUTTER
00004   1/2  TEASPOON SUGAR
00005   1/2  CUP FLOUR
00006   1    PINCH OF SALT
00007   2    EGGS
00008   2    CUPS HEAVY CREAM, WHIPPED
00009 CHOCOLATE SAUCE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00010
00011  12   OUNCES SEMI-SWEET CHOCOLATE
00012   2   OUNCES UNSWEETENED CHOCOLATE
00013   1   CUP HEAVY CREAM
00014   2   OUNCES COGNAC
00015
00016 * * * End of File * * *


====> merge :15 -/PUFFS/ 35
                                                            X E D I T  1 File
```



```
 DESSERT  COOKBOOK A1  F 80  Trunc=80 Size=9 Line=6 Col=1 Alt=1
12 lines merged



00000 * * * Top of File * * *
00001 CREAM PUFFS                       CHOCOLATE SAUCE
00002
00003   2    OUNCES BUTTER               12   OUNCES SEMI-SWEET CHOCOLATE
00004   1/2  TEASPOON SUGAR               2   OUNCES UNSWEETENED CHOCOLATE
00005   1/2  CUP FLOUR                    1   CUP HEAVY CREAM
00006   1    PINCH OF SALT                2   OUNCES COGNAC
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00007   2    EGGS
00008   2    CUPS HEAVY CREAM, WHIPPED
00009
00010 * * * End of File * * *






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

Figure 9. MERGE Subcommand — Before and After

Responses

The last line merged becomes the new current line.

Messages and Return Codes

498E Not executed--the two areas to merge overlap each other [RC=1]

520E	**Invalid operand:** *operand* **[RC=5]**
527E	**Invalid column number [RC=1]**
543E	**Invalid number:** *number* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
546E	**Target not found [RC=2]**
581E	**Subcommand is not valid in extended mode [RC=3]**
593E	**{No|*nn*} lines merged,** *nn* **line(s) {truncated|spilled} [RC=3]**

where return codes are:

0	Normal
1	Overlapping groups of lines; invalid column number
2	Target line not found
3	Truncated or spilled, or subcommand is not valid in extended mode
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

MODIFY (Macro)

```
►►──MODify──keyword──────────────────────────────────────────────────────────►◄
```

Purpose

Use the MODIFY macro to display a subcommand and its current operand values in the command line, so a new operand value can be typed over the current one and the subcommand immediately reentered.

Operands

keyword
is one of the following and is valid with the SET, QUERY, EXTRACT, or TRANSFER subcommands:

```
ALT                     HEX                     SCALe
APL                     IMage                   SCOPE
ARBchar                 IMPcmscp                SCReen
AUtosave                LASTLorc                SELect
BFSLINE                 LINENd                  SERial
BRKkey                  LRecl                   SHADow
CASE                    MACRO                   SIDcode
CMDline                 MASK                    SPAN
COLOR field¹            MSGLine                 SPILL
COLPtr                  MSGMode                 STAY
COLumn                  NAMETYPE                STReam
CTLchar                 NONDisp                 SYNonym
CTLchar char            NULls                   TABLine
CURLine                 NUMber                  TABS
DISPlay                 PAn                     TERMinal
ENTer                   PACK                    TEXT
ESCape                  PFn                     TOFEOF
ETARBCH                 PNAME                   TRunc
ETMODE                  PREfix                  VARblank
FILler                  PREfix Synonym name     Verify
FMode                   RANge                   VERShift
FName                   RECFm                   WRap
FType                   REMOte                  Zone
FULLread
```

[1]See “SET COLOR” on page 253 for a list of the fields.

Usage Notes

1. All of the above keywords cause the corresponding SET subcommand and its current operand value to be displayed, except for the following:

   `modify column`

   displays `CLOCATE :nn`, where: `nn` is the current column.

   `modify mask`

   displays the current mask, so you can type over it to modify it.

   `modify synonym`

   displays `SET SYNONYM ON` or `SET SYNONYM OFF`.
2. The COLOR operand may be specified as COLOR or COLOUR.

3. MODIFY CTLCHAR can be specified only when no control characters are defined. Otherwise, specify MODIFY CTLCHAR *char*.

Examples

This section shows examples of the MODIFY macro.

Example 1

The following example displays your current zone setting.

```
mod zone
```

```
SET ZONE 5 25
```

You can type over the 25 and change it to 50. Then press ENTER. The new zone settings are 5 and 50.

Example 2

The following example displays current nulls setting.

```
mod nulls
```

```
SET NULLS OFF
```

You can type over the OFF and change it to ON. Then press ENTER. The new setting is ON.

Responses

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

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
529E	**Subcommand is only valid in {display|editing} mode [RC=3]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
3	Subcommand valid only in display mode
5	Invalid or missing operand(s)
6	Subcommand rejected in the PROFILE due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

MOVE

```
►►──MOve──target1──target2──────────────────────────────────────────────►◄
```

Purpose

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.

Operands

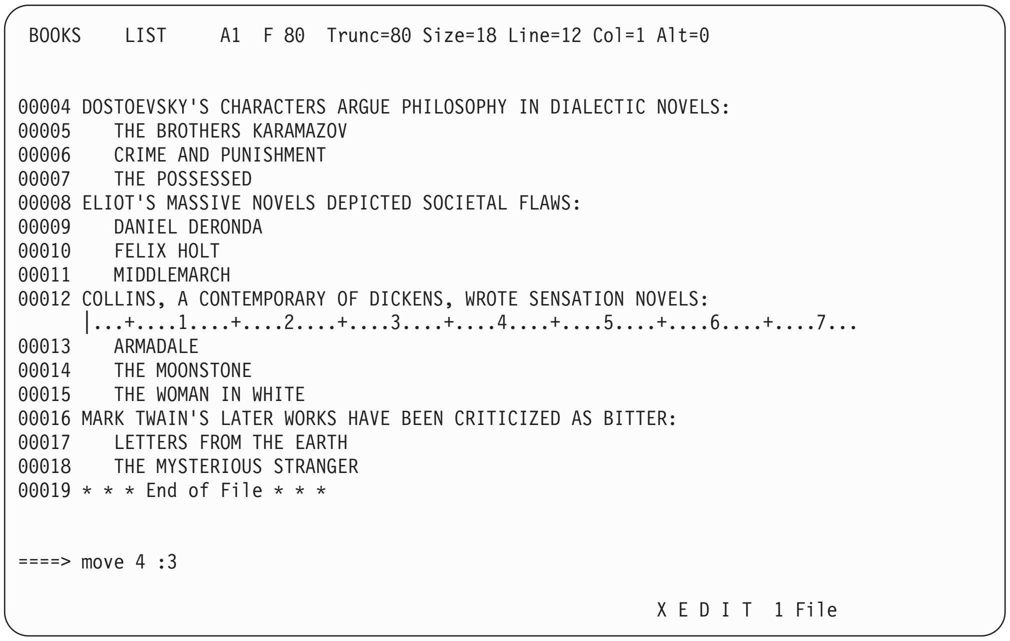
target1
: defines the number of lines to be moved. The lines are removed from the file starting with the current line up to, but not including, *target1*.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

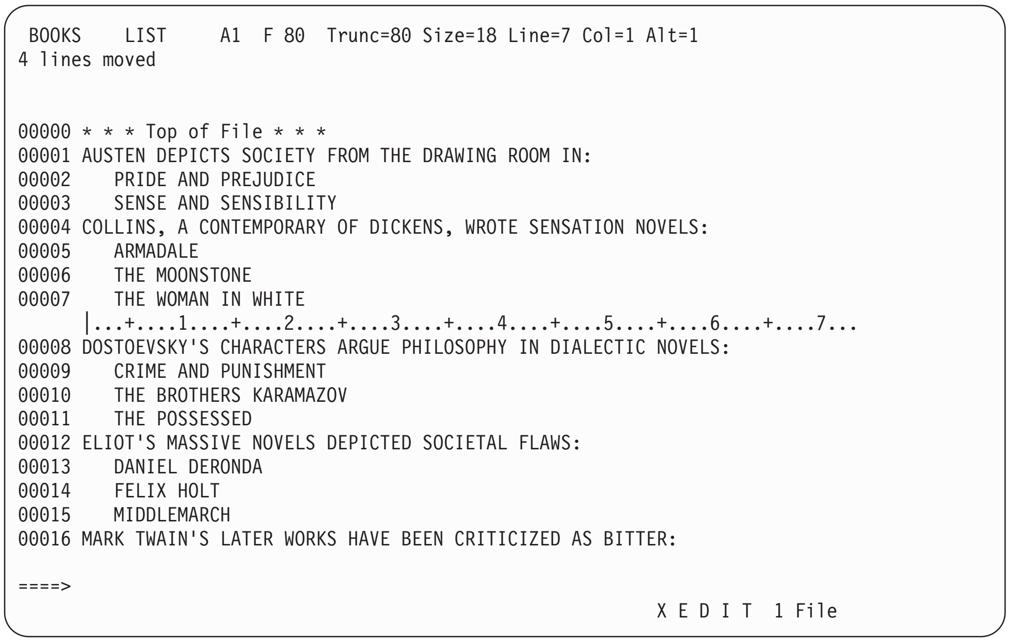
target2
: defines the destination line. The data is moved after *target2*.

Examples

Figure 10 on page 165 is a before-and-after example of the MOVE subcommand. Line 12 is the current line. For more information, see *z/VM: XEDIT User's Guide*.



```
 BOOKS    LIST     A1  F 80  Trunc=80 Size=18 Line=12 Col=1 Alt=0


00004 DOSTOEVSKY'S CHARACTERS ARGUE PHILOSOPHY IN DIALECTIC NOVELS:
00005    THE BROTHERS KARAMAZOV
00006    CRIME AND PUNISHMENT
00007    THE POSSESSED
00008 ELIOT'S MASSIVE NOVELS DEPICTED SOCIETAL FLAWS:
00009    DANIEL DERONDA
00010    FELIX HOLT
00011    MIDDLEMARCH
00012 COLLINS, A CONTEMPORARY OF DICKENS, WROTE SENSATION NOVELS:
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00013    ARMADALE
00014    THE MOONSTONE
00015    THE WOMAN IN WHITE
00016 MARK TWAIN'S LATER WORKS HAVE BEEN CRITICIZED AS BITTER:
00017    LETTERS FROM THE EARTH
00018    THE MYSTERIOUS STRANGER
00019 * * * End of File * * *


====> move 4 :3
                                                            X E D I T  1 File
```



```
 BOOKS    LIST     A1  F 80  Trunc=80 Size=18 Line=7 Col=1 Alt=1
4 lines moved


00000 * * * Top of File * * *
00001 AUSTEN DEPICTS SOCIETY FROM THE DRAWING ROOM IN:
00002    PRIDE AND PREJUDICE
00003    SENSE AND SENSIBILITY
00004 COLLINS, A CONTEMPORARY OF DICKENS, WROTE SENSATION NOVELS:
00005    ARMADALE
00006    THE MOONSTONE
00007    THE WOMAN IN WHITE
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00008 DOSTOEVSKY'S CHARACTERS ARGUE PHILOSOPHY IN DIALECTIC NOVELS:
00009    CRIME AND PUNISHMENT
00010    THE BROTHERS KARAMAZOV
00011    THE POSSESSED
00012 ELIOT'S MASSIVE NOVELS DEPICTED SOCIETAL FLAWS:
00013    DANIEL DERONDA
00014    FELIX HOLT
00015    MIDDLEMARCH
00016 MARK TWAIN'S LATER WORKS HAVE BEEN CRITICIZED AS BITTER:

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

Figure 10. MOVE Subcommand — Before and After

Responses

The last line moved becomes the new current line.

The editor displays the following message:

506I *nn* lines moved

Messages and Return Codes

505E Not executed--the target line (*nn*) is within the lines to move [RC=1]
520E Invalid operand: *operand* [RC=5]
545E Missing operand(s) [RC=5]

546E	Target not found [RC=2]

where return codes are:

0	Normal
1	Target line within the lines to move
2	Target line not found
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

MSG

```
           (1)
>>──MSG──┬──────┬─────────────────────────────────────────────────────><
         └─text─┘
```

Notes:

1 If *text* is not specified, a blank line is displayed.

Purpose

Use the MSG subcommand to display a message in the message area.

Operands

text
: is the text of the message to be displayed. If omitted, a blank line is displayed.

Usage Notes

1. The MSG subcommand does not sound the alarm. Use the EMSG subcommand to sound the alarm.
2. On the display terminal, the message is displayed in the message area of the logical screen. MSG can also be used to type a message on a typewriter terminal.
3. If you are in a GUI environment, messages that do not fit in the message area are displayed in the XEDIT Messages window.

Examples

In the following example, the MSG subcommand displays HAVE FUN NOW in the message line.

```
msg HAVE FUN NOW
```

Notes for Macro Writers

1. Use the MSG subcommand within a macro to display a message at the terminal.
2. When multiple messages are issued and they do not fit on the message line(s) as defined with SET MSGLINE, they are passed to CMS. If full-screen CMS is OFF, the message replaces the file image on the screen and the terminal is placed in a `MORE...` (waiting) status. To get the file image back on the screen, press CLEAR.

   If full-screen CMS is ON, and the CMS command displays text, the text appears in the CMSOUT window. (Text appears in the CMSOUT window by default; you can route the text to another window with the CMS VSCREEN ROUTE command, explained in *z/VM: CMS Commands and Utilities Reference*). To see all the text in the virtual screen, you can use the CMS WINDOW FORWARD or WINDOW BACKWARD command. The screen is cleared automatically when you scroll to the bottom of the virtual screen. Alternately, you can clear the screen with the CMS WINDOW DROP command. If you delete the CMSOUT window, you do not see messages passed to CMS.

Responses

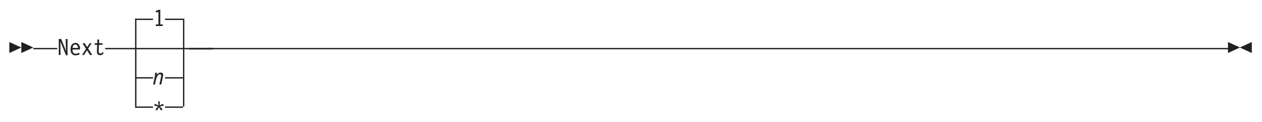
The message is displayed in the message area of the screen.

Messages and Return Codes

This subcommand issues no messages, it issues return codes as follows:

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

NEXT



Purpose

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.)

Operands

n is the number of lines to move the line pointer. If you omit *n*, the pointer moves down only one line.

***** the line pointer moves to the End of File line.

Usage Notes

1. The `NEXT` *n* subcommand is equivalent to a plus (+) target definition.

   For example:

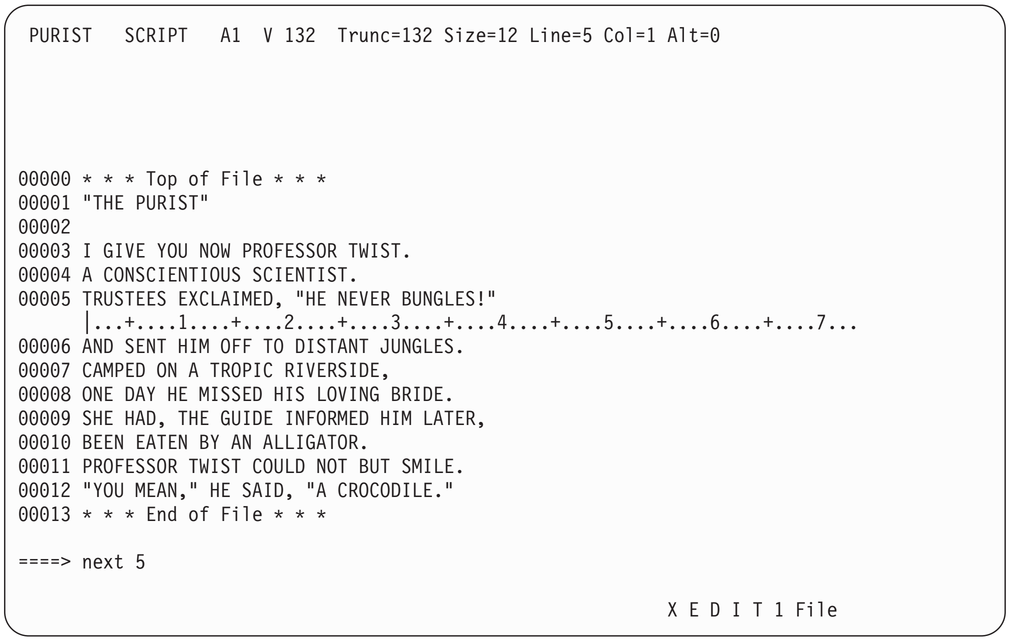
   ```
   next 3
   ```

   is equivalent to

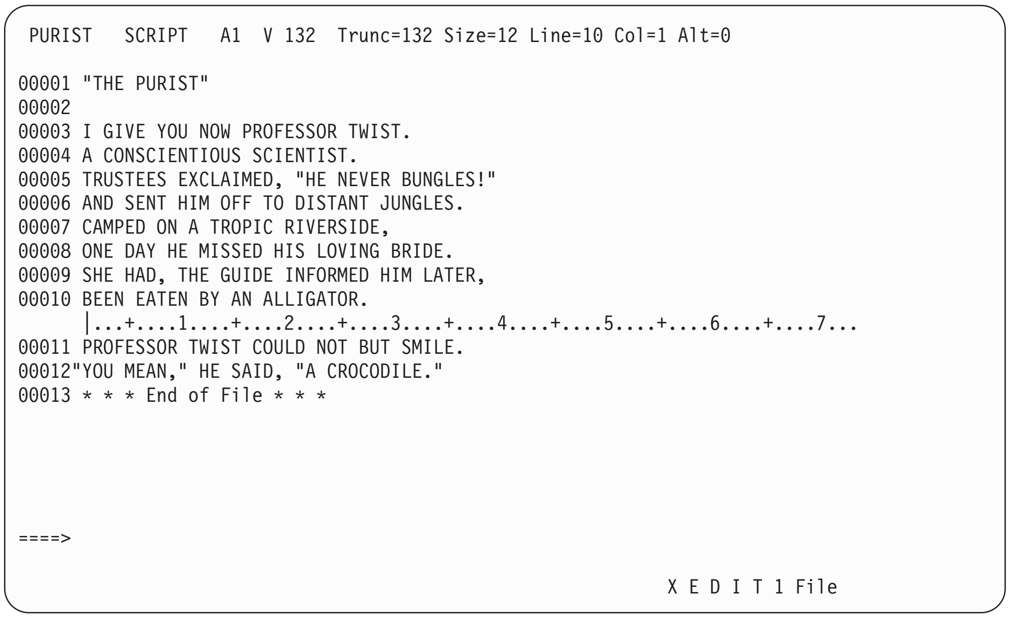
   ```
   +3
   ```

Examples

Figure 11 on page 170 is the before-and-after example of the NEXT subcommand. Line 5 is the current line.



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=5 Col=1 Alt=0




00000 * * * Top of File * * *
00001 "THE PURIST"
00002
00003 I GIVE YOU NOW PROFESSOR TWIST.
00004 A CONSCIENTIOUS SCIENTIST.
00005 TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00006 AND SENT HIM OFF TO DISTANT JUNGLES.
00007 CAMPED ON A TROPIC RIVERSIDE,
00008 ONE DAY HE MISSED HIS LOVING BRIDE.
00009 SHE HAD, THE GUIDE INFORMED HIM LATER,
00010 BEEN EATEN BY AN ALLIGATOR.
00011 PROFESSOR TWIST COULD NOT BUT SMILE.
00012 "YOU MEAN," HE SAID, "A CROCODILE."
00013 * * * End of File * * *

====> next 5

                                                              X E D I T 1 File
```



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=10 Col=1 Alt=0

00001 "THE PURIST"
00002
00003 I GIVE YOU NOW PROFESSOR TWIST.
00004 A CONSCIENTIOUS SCIENTIST.
00005 TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
00006 AND SENT HIM OFF TO DISTANT JUNGLES.
00007 CAMPED ON A TROPIC RIVERSIDE,
00008 ONE DAY HE MISSED HIS LOVING BRIDE.
00009 SHE HAD, THE GUIDE INFORMED HIM LATER,
00010 BEEN EATEN BY AN ALLIGATOR.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00011 PROFESSOR TWIST COULD NOT BUT SMILE.
00012"YOU MEAN," HE SAID, "A CROCODILE."
00013 * * * End of File * * *






====>

                                                              X E D I T 1 File
```

Figure 11. NEXT Subcommand — Before and After

Responses

The line pointed to becomes the new current line.

If you issue the NEXT subcommand while editing a file containing no records, the following message is displayed:

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

Messages and Return Codes

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

where return codes are:

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

NFIND

```
►►──NFind──text──────────────────────────────────────────────────────────►◄
```

Purpose

Use the NFIND subcommand to search forward in the file for the first line that does *not* start with the text specified in the operand. The search starts with the line after the current line.

Operands

text
is any text you do *not* want to find, beginning in column one of the next file line. Only nonblank characters in the operand are checked against the file.

Usage Notes

1. Use only one blank as a delimiter after the NFIND subcommand.
2. If SET IMAGE ON is in effect, tabs in the operand are changed to blanks (or filler characters) before the search, and the search begins in the first tab column.
3. To represent a blank in the operand, use an underscore character (_).
4. If you want to do the same type of search backward, see "NFINDUP" on page 173. See "FIND" on page 118 and "FINDUP" on page 120 to do searches which start with *text*.
5. If SET WRAP OFF is in effect, the search continues until the end of file.

   If SET WRAP ON is in effect, the search continues through the entire file and stops when the line where it started is reached again. When a WRAP occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```
6. The NFIND subcommand updates the LASTLORC buffer. See "SET LASTLORC" on page 281.

Responses

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

Messages and Return Codes

545E Missing operand(s) [RC=5]
586E Not found [RC=2]
592W Wrapped ....

where return codes are:

0	Normal
2	No target line was found
5	Missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

NFINDUP

```
►►──┬─NFINDUp─┬──text─────────────────────────────────────────────────────►◄
    └─NFUp────┘
```

Purpose

Use the NFINDUP subcommand to search *backward* in the file for the first line that does *not* start with the text specified in the operand. The search starts with the line before the current line.

Operands

text
is any text you do *not* want to find, beginning in column one of the file line before the current line. Only the nonblank characters in the operand are checked against the file.

Usage Notes

1. Use only one blank as a delimiter after the NFINDUP subcommand.
2. If SET IMAGE ON is in effect, tabs in the operand are changed to blanks (or filler characters) before the search, and the search begins in the first tab column.
3. To represent a blank in the operand, use an underscore character (_).
4. If you want to do the same type of search forward, see “NFIND” on page 172. See “FIND” on page 118 and “FINDUP” on page 120 to do searches which start with *text*.
5. If SET WRAP OFF is in effect, the search continues until the top of file.

   If SET WRAP ON is in effect, the search continues through the entire file and stops when the line where it started is reached again. When a WRAP occurs under these conditions, the following warning message is displayed:

   ```
   592W Wrapped ....
   ```
6. The NFINDUP subcommand updates the LASTLORC buffer. See “SET LASTLORC” on page 281.

Responses

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

Messages and Return Codes

545E Missing operand(s) [RC=5]
586E Not found [RC=2]
592W Wrapped ....

where return codes are:

0	Normal
2	No target line was found
5	Missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

OVERLAY

```
►►──Overlay──text──────────────────────────────────────────────────────►◄
```

Purpose

Use the OVERLAY subcommand to replace selectively one or more characters in the current line with the corresponding nonblank characters in the line being keyed in. If SET IMAGE ON is in effect, replacement starts at the first tab column of the current line.

Operands

text
: specifies an input line that replaces corresponding character positions in the current line.

Usage Notes

1. Blank characters in the input line indicate 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 also leave the corresponding characters in the current line unchanged.

   For example, the following subcommand adds a comment to an assembler language statement (file type ASSEMBLE) whose settings are defined by SET TABS 1 10 16 30 35 . . . :

   ```
   overlayTTTTcomment
   ```

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

Messages and Return Codes

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

where return codes are:

0	Normal
3	Truncated or spilled, or subcommand is not valid in extended mode
4	No line(s) changed

5	Missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

PARSE (Macro)

```
                     ┌──────────────┐
                     ▼              │
►►──PARSE──startcol───┬─Alphaword─┬─┴──────────────────────────────────────────►◄
                     ├─Dblstring─┤
                     ├─Line──────┤
                     ├─Number────┤
                     ├─String────┤
                     ├─Target────┤
                     └─Word──────┘
```

Purpose

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.

Operands

startcol
specifies the starting column in the input line where the parsing is to begin.

The following keywords define the sequence and types of the operands:

Alphaword
specifies the operand must be an alphabetic character string.

Dblstring
specifies 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.

Line
specifies the unprocessed part of the line being parsed. If the first character is a blank, it is excluded.

Number
specifies the operand must be a numeric character string.

String
specifies the operand must be a delimited string; the delimiter is the first character of the string.

Target
specifies the operand must be an XEDIT target.

Word
specifies the operand must be a character string, either alphabetic, numeric, or mixed, delimited by blanks.

Notes for Macro Writers

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

   ```
   push string
   ```

2. As a result of the PARSE macro, several lines are stacked last-in first-out (LIFO) in the console stack:

a. The first line contains an integer that indicates the number of stacked lines that follow. One line is stacked for each recognized operand in the parsed line and they are in the order they were specified.

b. Each of the remaining lines contains the starting column and the length of the operand, except for STRING and DBLSTRING operands.

   A line stacked for a STRING operand contains the:

   - starting column of the delimited string
   - length of the delimited string (including delimiters)
   - starting column of the string itself (without the delimiter)
   - length of the string (without delimiters)

   A line stacked for a DBLSTRING operand contains the:

   - starting column of the delimited strings
   - length of the delimited strings (including delimiters)
   - starting column of the first string itself (without the delimiter)
   - length of the first string (without delimiters)
   - starting column of the second string (without the delimiter)
   - 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.

Messages and Return Codes

This macro issues no messages, but it issues return codes as follows:

–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 completed parsing are available from the console stack.
5	Invalid operand.
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring.

POWERINP

```
►►──POWerinp──────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the POWERINP (power input) subcommand to enter an input mode 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 you press ENTER, the editor divides the data into file lines and puts together any split words.

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 ENTER. When you press ENTER, the last input line becomes the current line and is displayed at the top of the screen, and you can continue typing data.
3. If SET AUTOSAVE is in effect for your editing session, the file is automatically saved when the number of alterations equals or exceeds the number in the SET AUTOSAVE subcommand. However, no message telling you the file has been autosaved is displayed in power input mode.
4. To exit from power input mode, press ENTER without modifying the last displayed screen. The editor divides the data you typed into appropriate file lines. If necessary, lines are split at word boundaries and, when possible, the data is divided into lines that do not wrap.
5. PF keys cannot be used in power input mode. (Pressing a PF key is equivalent to pressing ENTER.) Any prefix subcommands or macros are not executed until you exit from power input mode.
6. If you want to cause a break in the data you type in power input mode, that is, you want data to start on a new line (for example, a new paragraph or SCRIPT/VS control words, which must start in column one), you can type a line-end character (see "SET LINEND" on page 282) before the data 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.#
   ```

   The data is entered in the file as:

   ```
   ===== .sp
   ===== A pound sign causes the data to start on a new line.
   ```

   Any leading blanks after the line-end character are eliminated when the data is entered in the file. The first nonblank character after the line-end character is placed in column one.
7. A word cannot be longer than the truncation setting.
8. You can use the insert key to insert characters in a line while in power input mode. When characters are inserted, the entire stream of data shifts to the right.

9. Any prefix subcommands or macros are not executed until you exit from power input mode.
10. If the width of your virtual screen or window is less than that of the physical screen, you cannot type continuously in power input mode. The keyboard locks when you come to the end of a line. To unlock the keyboard, press `RESET` and return the cursor to start a new line.
11. If you are in a GUI environment, you must tab to get to the next line when using POWERINP.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Responses

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

- The first line of the screen contains the file ID and the heading,

```
fname ftype fmode * * * P o w e r  T y p i n g * * *  Alt=n
```

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

Messages and Return Codes

117S	**Error writing to display terminal [RC=8]**
503E	**{Truncated|Spilled} [RC=3]**
529E	**Subcommand is only valid in {display|editing} mode [RC=3]**
557S	**No more storage to insert lines [RC=4]**
581E	**Subcommand is not valid in extended mode [RC=3]**

where return codes are:

0	Normal
3	Truncated or spilled; subcommand valid only for display terminal, or subcommand is not valid in extended mode
4	Insufficient storage available
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	I/O error

PRESERVE

```
►►──PREServe──────────────────────────────────────────────────────────────►◄
```

Purpose

Use the PRESERVE subcommand to save the settings of various XEDIT variables until a subsequent RESTORE subcommand is issued.

Usage Notes

1. The following settings are saved:
   a. Current LEFT or RIGHT
   b. The following SET subcommand options:

ARBCHAR	IMPCMSCP	SERIAL
AUTOSAVE	LASTLORC	SHADOW
BFSLINE	LINEND	SPAN
CASE	LRECL	SPILL
CMDLINE	MACRO	STAY
COLOR	MASK	STREAM
COLPTR	MSGMODE	SYNONYM
CURLINE	NAMETYPE	TABLINE
DISPLAY	NULLS	TABS
ESCAPE	NUMBER	TOFEOF
ETARBCH	PACK	TRUNC
FILLER	PNAME	VARBLANK
FMODE	PREFIX	VERIFY
FNAME	RECFM	WRAP
FTYPE	SCALE	ZONE
HEX	SCOPE	=
IMAGE		

   Note: Only the SET values are saved for CURLINE, not the values returned by the EXTRACT CURLINE. For example, the current line pointer is not saved.

2. The following values are *not* saved:

   Current line pointer

   Column pointer

   All SET subcommand options not listed in Usage Note 1b

Messages and Return Codes

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

where return codes are:

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

PURGE

```
►►──PURge──macro──────────────────────────────────────────────────────────►◄
```

Purpose

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

Operands

macro
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 uses EXECLOAD to load the macro into virtual storage. The macro remains in storage for the entire session. The macros the editor has loaded are removed (with EXECDROP) at the end of the session. (When storage becomes unavailable, the copy of the least-recently used macro is removed.) The PURGE subcommand is useful if you modify a macro and want the editor to read the new macro from disk or directory.

   Any time a macro that is being edited is filed or saved, an automatic PURGE is executed for that macro name. Therefore, PURGE is needed only after the CMS commands RENAME or DISK LOAD or with disk operations performed *outside* XEDIT.
2. When a macro invokes the PURGE subcommand and the PURGE is successfully completed, the return code is set to zero. If the macro to be purged is not in storage, the return code is set to 3. If the macro to be purged is in use, the return code is set to 4 and the macro is not purged.

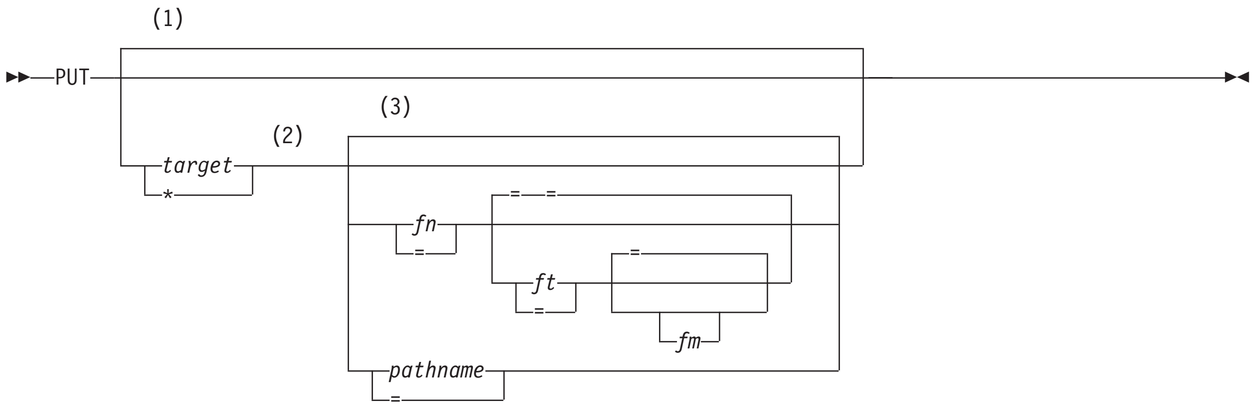
Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
578W	*Macro* **macro is not currently in storage [RC=3]**

where return codes are:

0	Normal
3	Macro is not currently in storage
4	Macro is in use, do not purge
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

PUT



Notes:

1 If *target* is not specified, the current line is inserted into a temporary SFS or minidisk file the editor creates.

2 The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the BFS format required for byte file system files (*pathname*).

3 The line(s) specified by *target* is placed in a temporary SFS or minidisk file the editor creates.

Purpose

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 you are creating; or a temporary SFS or minidisk file the editor creates. The original lines remain in the file you are editing.

Operands

target
defines the number of lines to be inserted into another file. Lines are inserted beginning with the current line, up to but not including the target line. If *target* is not specified, only the current line is inserted into a temporary file the editor creates. If you want to specify a filename, you *must* specify a target.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see “LOCATE” on page 147 and *z/VM: XEDIT User’s Guide*.

***** the rest of the file is inserted.

fn is the file name of the SFS or minidisk file into which lines are to be inserted. If the file does not exist, the editor creates it and displays the message, `Creating new file`, in the message area.

ft is the file type of the SFS or minidisk file into which lines are to be inserted. If *ft* is not specified, the editor uses the file type of the file you are currently editing.

fm is the file mode of the file into which lines are to be inserted. If *fm* is not specified, the editor uses the file mode of the file you are currently editing.

pathname
is the name of the byte file system (BFS) regular file into which the 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. See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name.

The length of the path name entered on this subcommand is limited. Use `SET PNAME` *`pathname_fragment1`* (or `PN` *`pathname_fragment1`*) and `SET PNAME APPEND` *`pathname_fragment2`* one or more times. Then use `PUT` *`target`* `=` if you wish to use a longer path name. Refer to Chapter 1, "Rules and Conventions," on page 1 for more information about the maximum lengths allowed for XEDIT subcommands.

Usage Notes

1. When NAMETYPE CMS is in effect, specifying an equal sign (=) for the *fn, ft,* and *fm* operands causes the corresponding FNAME, FTYPE, and FMODE values to be used. If not specified as an equal sign, the operand will be translated to uppercase.

   When NAMETYPE BFS is in effect, specifying an equal sign (=) for the *pathname* operand causes that value set using SET PNAME to be used. The path name is converted to uppercase only when SET CASE UPPER is in effect.

   If you change the NAMETYPE value, and then attempt to use an equal sign without setting the appropriate FNAME, FTYPE, or PNAME values, the editor returns an error. If you change the NAMETYPE value to CMS and then use an equal sign without setting FMODE, it defaults to file mode A1.
2. If the specified file (*fn ft fm*) exists, the lines are added to the end of the file. If the file is in an SFS file control directory, you must have write authority to the file; the directory can be accessed either read-only or read/write. However, if the directory is accessed in read-only mode and the CMS command SET RORESPECT ON has been issued, an attempt to PUT will fail. If the file is in an SFS directory control directory, you must have directory control write authority to the directory, and it must be accessed as read/write.
3. If the specified file (*fn ft fm*) does not already exist, the editor creates it and inserts the lines. If the file is in an SFS file control directory, you must have write authority to the directory; the directory can be accessed in either read-only or read/write mode. However, if the directory is accessed in read-only mode and the CMS command SET RORESPECT ON has been issued, an attempt to PUT will fail. If the file is in an SFS directory control directory, you must have directory control write authority to the directory, and it must be accessed as read/write.
4. If the specified BFS file (*pathname*) exists, the lines are added to the end of the file. You must have permission to write to the file. If it does not exist, the editor creates the BFS file and inserts the lines. You must have permission to write to the parent directory.
5. Permissions for a new BFS file are set based on the current value of the mask, with the exception of execute permissions, which are not set to ON. Use OPENVM PERMIT to change the defaults. See *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM for more information on these OPENVM commands.

6. The BFSLINE value determines whether end-of-line characters are inserted in the file. The RECFM value determines whether trailing blanks are stripped from the file. Refer to “SET BFSLINE” on page 245 and “SET RECFM” on page 319 for more information.
7. Use the QUERY DIRATTR command or the QUERY ACCESSED command to determine whether the SFS directory has the FILECONTROL or the DIRCONTROL attribute. (In the QUERY ACCESSED display, the *Vdev* column contains *DIR* for FILECONTROL directories and *DIRC* for DIRCONTROL directories.)
8. If you do not specify a file ID (*fn ft fm* or *pathname*), the lines specified by *target* are inserted into a temporary SFS or minidisk file the editor creates. These lines can be retrieved each time a subsequent GET subcommand is entered without an operand. This temporary file is replaced each time a successful PUT (or PUTD) subcommand is entered. It is erased when all rings are exited. Thus, entering a PUT subcommand without specifying a file ID is like storing lines in a temporary holding area. You can retrieve the temporary file without knowing its file ID.

   The editor uses a temporary SFS or minidisk file to preserve these lines even if the NAMETYPE setting is BFS.
9. When you are editing multiple files and multiple rings, only one temporary file is available. It may be used to insert data from one file into another.
10. If the specified file (*fn ft fm*) is packed, use the CMS COPYFILE command with the UNPACK option to unpack the file before entering the PUT (or PUTD) subcommand.
11. When you XEDIT a BFS file and attempt to write the file to a minidisk or SFS directory when the record format is V (variable), you will see an error message if the line length exceeds 65 535. (The maximum record length for a variable file in the CMS record file system is 65 535.)
12. The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the BFS format required for BFS files (*pathname*). NAMETYPE may be set using an XEDIT option, or by the SET NAMETYPE subcommand.
13. When a new BFS file is created, the owning UID established is the effective UID of the VM User ID on which the request was issued. The GID is the GID of the parent directory. Use OPENVM OWNER to change the defaults. Refer to the *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM OWNER for more information.

Examples

This section shows examples of the PUT subcommand. For more information, see *z/VM: XEDIT User’s Guide*.

Example 1

The following example stores the current line in a temporary file, to be inserted by a subsequent GET in another file.

```
put
```

Example 2

The following example stores from the current line up to the line containing *string*, for use by a subsequent GET in another file.

`put/`*string/*

Example 3

The following example creates a new file with lines from current line up to the *string*.

```
put  /string/  MY  NEWFILE
```

Example 4

The following example shows how to create a new BFS file with lines from the current line of an SFS or minidisk file up to the end of the file. You may also want to use the SET BFSLINE command to control the insertion of an end-of-line character.

```
set nametype bfs
put   *   /../VMBFS:filepoolid:filespaceid/pathname/filename
```

See page 124 for an example of how to use PUT and GET to transfer lines between files while editing multiple files.

Responses

1. If you are creating a file with the PUT subcommand, the following message is displayed.

   ```
   571I Creating new file:
   ```

2. The line following the last line PUT (the target line) becomes the new current line.

Messages and Return Codes

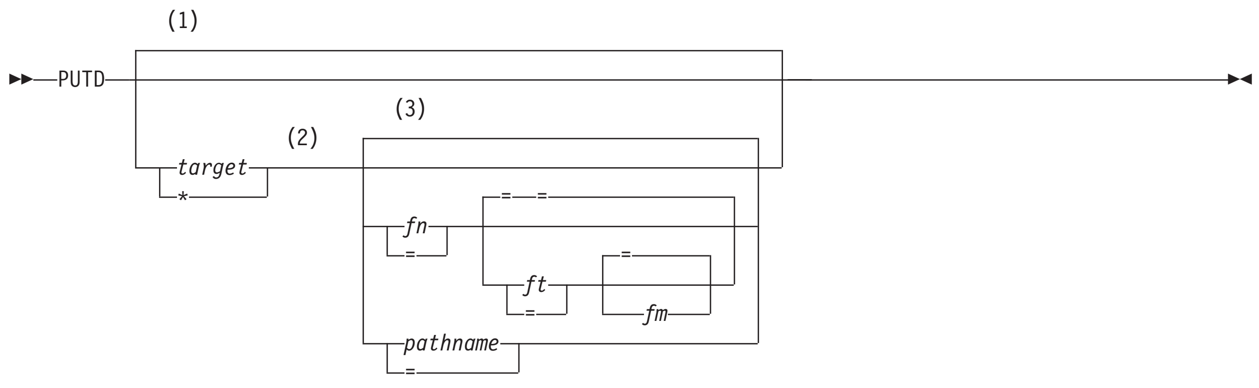
033E	**File is not a regular BFS file [RC=32]**
037E	**Base file for** *fn ft fm* **is in a DIRCONTROL directory accessed read-only. [RC=12]**
037E	**Filemode** *mode* **is accessed as read-only [RC=12]**
048E	**Invalid filemode** *mode* **[RC=24]**
054E	**Incomplete [or incorrect] fileid specified [RC=24]**
062E	**Invalid character in fileid {***fn ft fm*|*pathname***} [RC=20]**
069E	**Filemode** *mode* **not accessed [RC=36]**
104S	**Error** *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, or 100]**
105S	**Error** *nn* **writing file** *fn ft fm* **on disk or directory [RC=55, 70, 76, 99, or 100]**
229E	**Unsupported OS dataset, error** *nn* **[RC=80, 81, 82, 83, or 84]**
512E	**This is not allowed in CMS subset mode [RC=100]**
520E	**Invalid operand:** *operand* **[RC=5]**
531E	**BFS file space is full; clear some space**
531E	**Disk or file space is full; set new filemode or clear some space [RC=13]**
546E	**Target not found [RC=2]**
554E	**Not enough virtual storage available [RC=104]**
571I	**Creating new file:**
579E	**Records truncated to** *nn* **when added to** *fn ft fm* **[RC=3]**
698W	**New record length may result in loss of double-byte characters [RC=3]**
743E	**File** *fn ft fm* **is in an invalid format [RC=40]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code** *rc* **and reason code** *rs* **[RC=55, 104]**
1138E	**File sharing conflict for file {***fn ft fm*|*pathname***} [RC=70]**
1141W	**User filespace threshold exceeded**
1184E	**A directory is not found, or you are not permitted to use a directory in** *pathname* **[RC=28]**

1215E	**File** *fn ft fm* **is locked or in use by another user [RC=70]**
1258E	**Not {authorized|permitted} to write file {***fn ft fm***|***pathname***} [RC=12]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=55, 70, 76, 99, or 100]**
1301S	**Rollback error** *nn***, file** *fn ft fm* **left open**
2105E	**Permission is denied [24]**
2120E	**Unable to resolve path name** *pathname* **[RC=24]**
2131E	**[FNAME] [FTYPE] [PNAME] {is|are} not set and NAMETYPE {CMS|BFS} is in effect [RC=24]**
2134E	**Return code** *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname***] [RC=100]**
2154E	**File {***fn ft fm***|***pathname***} is migrated and implicit RECALL is set to OFF [RC=50]**
2155E	**DFSMS/VM error occurred during creation or recall of file {***fn ft fm***|***pathname***} [RC=51]**
2526E	**File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code** *acs rcode* **[RC=51]**

where return codes are:

0	Normal
1	TOF or EOF reached
2	Target not found
3	Records truncated
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Minidisk is read-only or not authorized or you do not have permission for the BFS file or byte file system mounted read-only
13	Minidisk or file space or BFS is full
20	Character in file name or file type or path name is not valid
24	File mode or path name is not valid
31	A rollback occurred
32	BFS file is not a regular file or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
40	File is packed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
84	Unsupported OS data set
99	A required system resource is not available
100	Error reading or writing file on disk or directory or byte file system
104	Insufficient storage available

PUTD



Notes:

1 If *target* is not specified, the current line is inserted into a temporary SFS or minidisk file the editor creates.

2 The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the BFS format required for byte file system files (*pathname*).

3 The line(s) specified by *target* is placed in a temporary SFS or minidisk file the editor creates.

Purpose

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 you are creating; or a temporary SFS or minidisk file the editor creates.

Unlike the PUT subcommand, the PUTD subcommand deletes the original lines from the file you are editing.

Operands

target
defines the number of lines to be inserted into another file. Lines are inserted beginning with the current line, up to but not including the target line. If *target* is not specified, only the current line is inserted.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

* the rest of the file is inserted.

fn is the file name of the SFS or minidisk file into which lines are to be inserted. If the file does not exist, the editor creates it and displays the message, `Creating new file`, in the message line.

ft is the file type of the SFS or minidisk file into which lines are to be inserted. If *ft* is not specified, the editor uses the file type of the file you are currently editing.

fm is the file mode of the file into which lines are to be inserted. If *fm* is not specified, the editor uses the file mode of the file you are currently editing.

pathname
is the name of the byte file system (BFS) regular file into which the 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. See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name.

The length of the path name entered on this subcommand is limited. Use `SET PNAME` *`pathname_fragment1`* (or `PN` *`pathname_fragment1`*) and `SET PNAME APPEND` *`pathname_fragment2`* one or more times. Then use `PUTD` *`target`* `=` if you wish to use a longer path name. Refer to Chapter 1, "Rules and Conventions," on page 1 for more information about the maximum lengths allowed for XEDIT subcommands.

Usage Notes

1. The PUTD subcommand, unlike the PUT subcommand, deletes the original lines from the file.
2. For additional information, see "Usage Notes" on page 183 of the PUT subcommand.

Examples

See "Examples" on page 184 of the PUT subcommand.

Responses

If you are creating a file with the PUTD subcommand, the following message is displayed:

```
571I Creating new file:
```

After lines are deleted from the original file, the line immediately following the last line deleted becomes the new current line. If lines are deleted in a backward direction, toward the top of the file, the line preceding the last deleted line becomes the new current line.

Messages and Return Codes

033E File is not a regular BFS file [RC=32]
037E Filemode *mode* **is accessed as read/only [RC=12]**
048E Invalid filemode *mode* **[RC=24]**
054E Incomplete [or incorrect] fileid specified [RC=24]
062E Invalid character in fileid {*fn ft fm***|***pathname***}**
069E Filemode *mode* **not accessed [RC=36]**
104S Error *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, or 100]**
105S Error *nn* **writing file** *fn ft fm* **on disk or directory [RC=55, 70, 76, 99, or 100]**
229E Unsupported OS dataset, error *nn* **[RC=80, 81, 82, 83, or 84]**
512E This is not allowed in CMS subset mode [RC=100]
520E Invalid operand: *operand* **[RC=5]**
531E BFS file space is full; clear some space
531E Disk or file space is full; set new filemode or clear some space
546E Target not found [RC=2]
554E Not enough virtual storage available [RC=104]
559W Warning: file is empty [RC=1]
571I Creating new file:

579E	**Records truncated to** *nn* **when added to** *fn ft fm* **[RC=3]**
698W	**New record length may result in loss of double-byte characters [RC=3]**
743E	**File** *fn ft fm* **is in an invalid format [RC=40]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code** *rc* **and reason code** *rs* **[RC=55, 104]**
1138E	**File sharing conflict for file {***fn ft fm***|***pathname***} [RC=70]**
1141W	**User filespace threshold exceeded**
1184E	**A directory is not found, or you are not permitted to use a directory in** *pathname* **[RC=28]**
1215E	**File** *fn ft fm* **is locked or in use by another user [RC=70]**
1258E	**Not {authorized|permitted} to write file {***fn ft fm***|***pathname***} [RC=12]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=55, 70, 76, 99, or 100]**
1301S	**Rollback error** *nn***, file** *fn ft fm* **left open**
2105E	**Permission is denied [24]**
2120E	**Unable to resolve path name** *pathname* **[RC=24]**
2131E	**[FNAME] [FTYPE] [PNAME] {is|are} not set and NAMETYPE {CMS|BFS} is in effect [RC=24]**
2134E	**Return code** *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname***] [RC=100]**
2154E	**File {***fn ft fm***|***pathname***} is migrated and implicit RECALL is set to OFF [RC=50]**
2155E	**DFSMS/VM error occurred during creation or recall of file {***fn ft fm***|***pathname***} [RC=51]**
2526E	**File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code** *acs rcode* **[RC=51]**

where return codes are:

0	Normal
1	TOF or EOF reached
2	Target not found
3	Records truncated
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Minidisk is read-only or not authorized or you do not have permission for the BFS file or byte file system mounted read-only
13	Minidisk or file space or BFS is full
20	Character in file name or file type or path name is not valid
24	File mode or path name is not valid
31	A rollback occurred
32	BFS file is not a regular file or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
40	File is packed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released

81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
84	Unsupported OS data set
99	A required system resource is not available
100	Error reading or writing file on disk or directory or byte file system
104	Insufficient storage available

QUERY

```
►►──Query──operand──────────────────────────────────────────────────►◄
```

Purpose

Use the QUERY subcommand to display in the message area the current setting of various editing options. You can specify only one option in each QUERY subcommand.

Operands

operand
may be any one of the following keywords. A complete description of each operand follows. For additional information on some of the operands, refer to its corresponding SET subcommand (if applicable).

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

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

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

APL
displays ON or OFF as defined in the SET APL subcommand.

ARBchar
displays ON or OFF and the arbitrary character specified in the SET ARBCHAR subcommand.

AUtosave
displays the current setting defined in the SET AUTOSAVE subcommand: the AUTOSAVE count, file ID, and number of alterations.

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

BFSLine
displays *lrecl*, NL, CRLF, CRNL, /*string*/, or /*hexstring* / as defined in the SET BFSLINE subcommand.

BRKkey
returns ON or OFF as defined by the terminal brkkey setting. If ON, then it also returns the PF or PA key currently set to be the BRKKEY.

CASE
displays the case setting U or M and R or I as defined in the SET CASE subcommand.

CMDline
displays ON, OFF, TOP, or BOTTOM as defined in the SET CMDLINE subcommand.

COLOR

```
►►──Query──COLOR──┬─field─┬──────────────────────────────────────────►◄
                  └─*─────┘
```

displays the current setting defined in the SET COLOR subcommand for field, color, extended highlighting, HIGH or NOHIGH, and programmed symbol set for the field requested or for all fields (when * is specified). The fields that can be specified are: Arrow, Cmdline, CUrline, Filearea, Idline, Msgline, Pending, PRefix, Scale, SHadow, STatarea, Tabline, and TOfeof. Note that this operand may be specified as COLOR or COLOUR.

COLPtr
displays ON or OFF as defined in the SET COLPTR subcommand.

COLumn
displays the column number of the column pointer.

CTLchar

```
►►──Query──CTLchar──┬──────┬─────────────────────────────────────────►◄
                    └─char─┘
```

If no character is specified (Q CTLCHAR), displays the escape character and all control characters defined in 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 (protected/unprotected field), attribute2 (color), attribute3 (extended highlighting),` `attribute4 (highlighting)`, and `attribute5 (programmed symbol set)`. If no attributes were defined for the character, displays `CTLCHAR char`.

CURLine
displays the line number of the current line as specified in the SET CURLINE subcommand.

CURSor
displays the current position of the cursor on the screen (line number and column number) and the position of the cursor in the file (line number and column number). If the cursor is not in the file area, two negative numbers (–1) are displayed for the position of the cursor in the file. The top and bottom of the range are considered to be in the file.

The current position of the cursor is the location where the cursor would be placed if the screen were displayed at this time. The current position reflects relative changes due to additions/deletions of lines resulting from prefix execution. It does not necessarily reflect where the cursor will eventually be located when the screen is actually displayed. This is because cursor priority cannot be resolved until the screen is displayed.

DISPlay
displays the range of selection levels that are included in the display, as specified in the SET DISPLAY subcommand.

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

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

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

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

ENTer
displays BEFORE, AFTER, ONLY, or IGNORE and the ENTER key definition as set in the SET ENTER subcommand.

EOF
displays ON or OFF as the editor determines. EOF is ON when the line pointer reaches end of file (or end of range).

EOL
displays ON or OFF as the editor determines. EOL is ON when the column pointer reaches zone2+1.

EPName
displays the byte file system (BFS) path name at the time the file was loaded. Because a path name can have leading or trailing quotes, single quotes are used to surround the path name. A path name of '' is displayed if the file that was loaded was not a BFS file.

Depending upon the path name format used when the file was loaded, this can be a relative path name, an absolute path name, or a fully qualified path name.

See “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the different forms of the BFS path name syntax.

ESCape
displays ON or OFF and the escape character defined in the SET ESCAPE subcommand.

ETARBCH
displays ON or OFF and the extended arbitrary character defined in the SET ETARBCH subcommand.

ETMODE
displays ON or OFF as defined in the SET ETMODE subcommand.

FILler
displays the filler character defined with the SET FILLER subcommand.

FMode
displays the two-character file mode.

FName
displays the eight-character file name.

FType
displays the eight-character file type.

FULLread
displays ON or OFF as defined in the SET FULLREAD subcommand.

GUI
displays ON or OFF to indicate whether GUI is active for the ring.

HEX
displays ON or OFF as specified in the SET HEX subcommand.

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

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

LASTLorc
displays the current contents of the last locate or change buffer, as SET LASTLORC or the editor specifies.

LASTmsg
displays the last message the editor issued. This message may or may not have been displayed, depending on the SET MSGMODE subcommand operands.

LENgth
displays the length of the current line from column one through the truncation column (excluding trailing blanks). The length is zero for Top of File and End of File lines.

LIBName
displays `LIBNAME file name` when MEMBER is ON. The file name is the library file name specified on the XEDIT or LOAD command. When MEMBER is OFF, the library file name is blank.

LIBType
displays `LIBTYPE file type` when MEMBER is ON. The file type is the library file type specified on the XEDIT or LOAD command. When MEMBER is OFF, the library file type is blank.

LIne
: displays the current line number, relative to the beginning of the file.

LINENd
: displays ON or OFF and the line-end character as defined in the SET LINEND subcommand.

LRecl
: displays the logical record length of the file.

LScreen
: displays six integers:
  - The number of lines and the number of columns in the logical screen
  - The line number and the column number defining the top left corner of the logical screen on the virtual screen
  - The number of lines and number of columns in the virtual screen.

MACRO
: displays ON or OFF as specified in the SET MACRO subcommand.

MASK
: displays the current mask line as defined in the SET MASK subcommand.

MEMber
: displays MEMBER ON when editing a member of a CMS library or MEMBER OFF when not editing a library member.

MSGLine
: displays ON or OFF, the location of the message line, the number of lines to which the message line can expand, and OVERLAY, if that has been specified, as defined in the SET MSGLINE subcommand.

MSGMode
: displays ON and LONG or SHORT as defined in the SET MSGMODE subcommand; if SET MSGMODE OFF is in effect, nothing is displayed.

NAMetype
: displays CMS or BFS as defined in the SET NAMETYPE subcommand.

NBFile
: displays the number of files you are currently editing.

NONDisp
: displays the character defined in the SET NONDISP subcommand.

NULls
: displays ON or OFF as specified in the SET NULLS subcommand.

NUMber
: displays ON or OFF as specified in the SET NUMBER subcommand.

PA

```
           ┌─PA*──┐
►►──Query──┤      ├──────────────────────────────────────────────────────►◄
           └─PA─n─┘
```

displays BEFORE, AFTER, ONLY, or IGNORE and if you specify:
- A particular key (PA*n*), QUERY PA displays the setting of that key.
- An asterisk or no argument at all, QUERY PA displays all PA key definitions.

PACK
displays ON or OFF as specified in the SET PACK subcommand.

PENDing

```
►►──Query──PENDing──┬───────┬──┬─────────┬──┬─name─┬──────────────────────►◄
                    └─BLOCK─┘  └─OLDNAME─┘  └─*────┘
```

displays the first entry in the pending list with “name” name or all the entries in the pending list (if * specified).

BLOCK

indicates that the pending list is to be checked for BLOCK entries only.

OLDNAME

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

name

indicates the name of the prefix subcommand or macro for which you are searching. If OLDNAME is also specified, *name* must be the original name of the prefix subcommand or macro, regardless of whether a synonym has been assigned to that name. Otherwise it is assumed to be a synonym (that is, a new name) or a name without a synonym.

*

indicates displaying all of the entries in the pending list. If BLOCK is also specified, * indicates displaying only the block entries.

The information is returned in the following form:

```
Line n : 'name', Oldname='name', OP1='x', OP2='y', OP3='z'
```

PF

```
           ┌─PF──*─┐
►►──Query──┤       ├──────────────────────────────────────────────────────►◄
           └─PF──n─┘
```

displays BEFORE, AFTER, ONLY, or IGNORE and if you specify:

- A particular key (PF*n*), QUERY PF displays the setting of that key.
- An asterisk or no argument at all, QUERY PF displays all PF key definitions.

Note: If you are editing a file in line mode (SET TERMINAL TYPEWRITER), XEDIT does not recognize which key caused an attention interrupt. The definition that is executed is the CP definition of that key. To QUERY keys in line mode, use the CP QUERY PF*nn* command.

PName
displays the current active byte file system (BFS) path name. Because a path name can have leading or trailing quotes, single quotes are used to surround

the path name. The total path name can be up to 1023 characters in length (not including surrounding quotes). A path name of '' is displayed if the file that was loaded was not a BFS file and SET PNAME has not been entered.

Depending upon the path name format used on SET PNAME or when the file was loaded, this can be a relative path name, an absolute path name, or a fully qualified path name. See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name syntax.

Point

```
►►──Query──Point──┬───┬────────────────────────────────────────────►◄
                  └─*─┘
```

QUERY POINT displays the symbolic name(s) associated with the current line, or displays a blank string if no names are specified for this line. QUERY POINT * displays all symbolic names that have been defined, starting at the top of the file, including the line number to which each name applies.

PREfix

```
►►──Query──PREfix──┬──────────────────────┬────────────────────────►◄
                   └─Synonym──┬─name─┬────┘
                              └─*────┘
```

QUERY PREFIX displays:

- ON, OFF, or NULLS
- RIGHT or LEFT

as specified in the SET PREFIX subcommand.

QUERY PREFIX SYNONYM * displays both the old and the new names of the synonyms defined for the prefix subcommand(s) or macro(s).

QUERY PREFIX SYNONYM *name* displays the synonym for the specified prefix subcommand or macro and its associated old name.

RANge
displays the line numbers of the top and bottom of the range defined in the SET RANGE subcommand.

RECFm
displays the record format F, V, FP, or VP, defined in the SET RECFM subcommand.

REMOte
displays ON or OFF depending upon whether a remote terminal is being used or upon the setting specified in the SET REMOTE subcommand.

RESERved
displays (on one line) the line numbers of the screen lines currently reserved. The line numbers are displayed in the order in which they were reserved.

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

The editor displays the following message:

```
530I nn file(s) in storage
```

The following information is displayed for each file:

```
fn ft fm recfm lrecl Trunc=truncno Size=sizeno
Line=lineno Col=colno Alt=altcount
```

where:

`fn` is the file name of the file.

`ft` is the file type of the file.

`fm` is the file mode of the file.

`recfm` is the record format. F is fixed-length records. V is variable-length records. FP is fixed-length packed. VP is variable length packed.

`lrecl` the logical record length of the largest record permitted in the file.

`truncno`
is the truncation column.

`sizeno` is the number of records currently in the file.

`lineno` is the position in the file of the current line.

`colno` is the column number in which the column pointer is located.

`altcount`
is the number of alterations since the last autosave or the number set with SET ALT.

SCALe
displays ON or OFF and the position of the SCALE as specified in the SET SCALE subcommand (or SCALE prefix subcommand).

SCOPE
displays DISPLAY or ALL as specified in the SET SCOPE subcommand.

SCReen
displays the attributes of the screens that have been defined with the SET SCREEN subcommand, preceded by SIZE, WIDTH, or DEFINE.

SELect
displays the selection level of the current line and the maximum selection level for the file as specified in the SET SELECT subcommand.

Seq8
displays OFF if the XEDIT command or LOAD subcommand was issued with the NOSEQ8 operand; if not, displays ON.

SERial
displays the serial identification, the increment value, and the serial number starting value as defined in the SET SERIAL subcommand.

SHADow
displays ON or OFF as specified in the SET SHADOW subcommand.

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

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

SPAN
displays ON or OFF, B or N, and *n* as defined in the SET SPAN subcommand.

SPILL
displays ON, OFF, or WORD as defined in the SET SPILL subcommand.

STAY
displays ON or OFF as specified in the SET STAY subcommand.

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

SYNonym

```
►►──Query──SYNonym──┬────────┬──────────────────────────────────────────────►◄
                    ├─name───┤
                    └─*──────┘
```

QUERY SYNONYM displays ON or OFF as specified in the SET SYNONYM subcommand.

QUERY SYNONYM * displays for each defined synonym its name, its minimum abbreviation, and the associated synonym definition, which includes the LINEND *char*, if specified, and everything that was specified in the SET SYNONYM subcommand after the size of the minimum abbreviation.

QUERY SYNONYM *name* displays the synonym, its minimum abbreviation, and the associated synonym definition, which includes the LINEND *char*, if specified, and everything that was specified in the SET SYNONYM subcommand after the size of the minimum abbreviation.

(If no synonym has been defined, only the name is displayed.) For example:

```
set synonym up 1 down
query syn u
```

displays the following: `SYNONYM UP U down`

TABLine
displays ON or OFF and the position of the TABLINE as defined in the SET TABLINE subcommand (or TABL prefix subcommand).

TABS
displays the tab column numbers defined in the SET TABS subcommand.

TARGet
displays the following information about the character string that matches the last target located with a LOCATE or CLOCATE subcommand: line and column number of the first character in the string; line and column number of the last character in the string. If the last target located was specified with *&*, then only information about the last string found is displayed.

If the target of the LOCATE or CLOCATE has been specified as an absolute line number, a relative displacement from the current line, or a line name, then QUERY TARGET displays the line number and current column position (twice).

Information that QUERY TARGET returns may be incorrect unless the QUERY is done immediately following the LOCATE or CLOCATE of the target. Any XEDIT subcommand issued between the LOCATE or CLOCATE of the target and the QUERY has the potential to invalidate the TARGET information.

TERMinal
displays DISPLAY or TYPEWRITER as defined in the SET TERMINAL subcommand.

TEXT
displays ON or OFF as specified in the SET TEXT subcommand.

TOF
displays ON or OFF as the editor determines. TOF is ON when the line pointer reaches top of file (or top of range).

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

TOL
displays ON or OFF as the editor determines. TOL is ON when the column pointer reaches zone1–1.

TRANSLat
displays ON or OFF, depending on whether the user has defined pairs of uppercase translate characters using the SET TRANSLAT subcommand.

TRunc
displays the truncation column number as defined in the SET TRUNC subcommand.

UNIQueid
displays a unique identifier associated with the file. The identifier has the form *rrrnnnnn* where *rrr* is the number XEDIT associates with the ring and *nnnnn* is the current autosave number. Note that when the ring number, *rrr*, is less than 100, leading zeros are dropped. The uniqueid is also used as the file name for the AUTOSAVE file.

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

UPDate
displays ON or OFF as the editor determines. Update is ON when the XEDIT command or LOAD subcommand has been issued and the UPDATE option was specified or implied.

VARblank
displays ON or OFF as specified in the SET VARBLANK subcommand.

Verify
displays H (if SET VERIFY with the HEX option was previously issued), the verification columns and ON or OFF as specified in the SET VERIFY subcommand.

VERShift
displays *n* or *–n*, which is the relative position of the screen over the file, as a result of any LEFT or RIGHT subcommands.

Width
displays the WIDTH value specified in the XEDIT command or LOAD subcommand.

WRap
displays ON or OFF as specified in the SET WRAP subcommand.

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

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

Usage Notes

1. The following options are not valid when queried from a typewriter terminal:

```
CMDLINE                    LSCREEN
CURSOR                     SCREEN
```

   If you query any of these operands from a typewriter terminal, the following message is displayed:

```
520E Invalid operand: operand
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
525E Invalid {PFkey|PFkey/PAkey} number [RC=5]
538E No name defined [RC=3]
545E Missing operand(s) [RC=5]
554E Not enough virtual storage available [RC=104]
586E Not found [RC=2]

where return codes are:

0	Normal
2	'QUERY PENDing' was issued, but no pending name or list is defined
3	'QUERY POINT *' was issued, but no symbolic names are defined.
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
104	Insufficient storage

QUIT

```
►►──QUIT──┬───┬──────────────────────────────────────────────────────►◄
          └─n─┘
```

Purpose

Use the QUIT subcommand or QQUIT to terminate the current editing session and leave the previous copy of the file, if any, intact on the disk or directory. If the file has been changed during the editing session, the editor displays a warning message asking you to confirm you want to issue a QUIT (instead of a FILE) subcommand. When issued from a macro, the QUIT subcommand can specify a return code.

Operands

n is a return code that may be specified when QUIT is issued from a macro.

Usage Notes

1. You can use the QUIT subcommand when you edit a file merely to examine, but not to change, its contents, or whenever you discover you have made errors in editing a file and want to cancel your editing session.
2. If only one file was edited, control is returned to the environment that invoked the editor.
3. If more than one file was being edited, only the current file is terminated. Control remains in the edit environment. You can use the CANCEL macro to *quit* multiple files.
4. The QUIT subcommand is initially assigned to the PF3 key.
5. When editing two or more members of a MACLIB in member mode, the lock on the library is not deleted until the last member that had the lock option in effect exits the ring.

Notes for Macro Writers

1. QUIT is defined as a synonym to PQUIT. The PQUIT (protected quit) subcommand causes a warning message to be displayed (see responses) if the file was changed during editing. To bypass the message and have the QUIT subcommand executed directly, you can define QUIT as a synonym to COMMAND QUIT, or you can issue QQUIT, which is an unprotected *quick quit.*

   The PQUIT subcommand clears the program stack. If you do not want the program stack to be cleared, use COMMAND QUIT.
2. If QUIT is issued from a macro, control remains in the macro until the macro finishes executing. Then, control is returned to the editor.

   If multiple files were being edited, only the current file is terminated.

   If QUIT executes successfully from a macro, the return code is set to *n* if specified. If *n* was not specified and only one file was being edited, a QUIT issued from a macro sets the return code to 1 and executes the QUIT when the macro completes execution. After issuing the QUIT, any subcommands issued in the macro result in a return code of 6.
3. QUIT or PQUIT may not be issued from a prefix macro.

4. Negative numbers may not be specified for *n*.

Responses

If only *one* file was edited, the CMS ready message indicates 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; type QQUIT to quit anyway
```

If you wish to save the changes, issue a FILE subcommand.

On a typewriter terminal, the following message is displayed:

```
553I Editing file: fn ft fm
```

Messages and Return Codes

509E	*Subcommand* **subcommand not valid from a prefix macro [RC=4]**
520E	**Invalid operand:** *operand* **[RC=5]**
543E	**Invalid number:** *number* **[RC=5]**
553I	**Editing file:** *fn ft fm*
554E	**Not enough virtual storage available**
577E	**File has been changed; type QQUIT to quit anyway [RC=12]**
622E	**Insufficient free storage for MSGLINE**
1300E	**Error** *nn* **unlocking file** *fn ft dirname*

where return codes are:

0	Normal and more than one file is being edited or more files currently being added in the ring
1	Only one file was edited
4	Invalid when issued from a prefix macro
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	File has been changed. Type QQUIT to QUIT anyway
n	The number specified as an operand to the subcommand

READ

```
            ┌─Cmdline────────────────────┐  ┌─Notag─┐
►►──READ────┼────────────────────────────┼──┼───────┼──────────────────────────────────────────►◄
            └──┬─All──────┬──┬────────┬──┘  └─Tag───┘
               └─Nochange─┘  └─Number─┘
```

Purpose

Use the READ subcommand to place information from the terminal in the console stack (LIFO). The READ subcommand is designed to be issued from a macro. After a READ subcommand is executed, a REXX macro has access to the stacked information through the REXX PULL instruction. (In an EXEC 2 macro, access can be gained through the &READ control statement.)

Operands

Cmdline
indicates stacking only the command input area in the console stack. This is the default.

All
indicates stacking all lines changed on the screen in the console stack, and stacking the command input area as the last line. The corresponding changes are also made to the copy of the file that is in virtual storage.

Nochange
is similar to the ALL option except the changes made on the screen are not made to the file being edited but are available from the console stack.

Number
indicates the file lines changed on the screen are to be prefixed by their file line numbers. Prefix area changes will be prefixed by their associated file line numbers only if the TAG option is specified.

Tag
indicates you want to insert a tag identifying the origin of the line at the beginning of each line stacked. (The tags are described in Note 3 on page 205).

Notag
specifies no tags are inserted in the stacked lines.

Notes for Macro Writers

1. If a READ is issued from a typewriter terminal, only the command line is placed in the stack.
2. Pressing ENTER, a PF key, or a PA key terminates a READ subcommand; however, a key assigned to COPYKEY, NULLKEY, TABKEY, or the CP BRKKEY does not terminate a READ. Pressing CLEAR clears any data typed since the last time the ENTER key, a PF key, or a PA key was pressed, but this also does not terminate a READ. The operand specified for the READ subcommand (CMDLINE, ALL, or NOCHANGE) determines which lines are placed in the console stack. The same number of lines are stacked whether TAG or NOTAG is specified. The TAG operand causes each line in the stack to be preceded by a tag. The various tags are described in Note 3.

Using READ CMDLINE

The key pressed to terminate the READ command (a PF key, a PA key, or the ENTER key) and how that key is defined (BEFORE, AFTER, ONLY, or IGNORE) determine which lines are placed in the console stack.

If something was entered on the command line, the console stack contains the following line(s), as determined by the definition of the key that was pressed:

If the key was defined as BEFORE, the console stack contains:

a. ENTER, PF, or PA key value (depending on which key was pressed)
b. Command line

If the key was defined as AFTER, the console stack contains:

a. Command line
b. ENTER, PF, or PA key value (depending on which key was pressed)

If the key was defined as IGNORE, the console stack contains the command line.

If the key was defined as ONLY, the console stack contains the ENTER, PF, or PA key value (depending on which key was pressed).

If nothing was entered on the command line (or ERASE EOF was used to clear the command line), the console stack contains the definition of the key (ENTER, PF, or PA key) that was pressed.

Using READ ALL or READ NOCHANGE

The editor scans the screen (from top to bottom, left to right) and stacks all modified fields. Each modified field is stacked as a separate line. With either the ALL or the NOCHANGE operand, the console stack contains:

a. ENTER, PF, or PA key value (depending on which key was pressed)
b. Lines and prefix areas changed on the screen (if any)
c. Command line (if something was entered on the command line)

If nothing was changed on the screen, the console stack contains the ENTER, PF, or PA key value (depending on which key was pressed).

Using CTLCHARs in non-RESERVED lines with READ NOCHANGE, may cause unpredictable results.

3. With the TAG operand specified, a tag that identifies the field precedes each line in the stack. The tags and their meanings are as follows:

   CMD - command line
   ETK - ENTER key
   FIL - file line
   PAK - PA key
   PFK - PF key
   PRF - prefix area
   RES - reserved line

   The tag is followed by additional information and the modified field itself:

   - `CMD string`

     where `string` is whatever was typed in the command line.

     The `string` can be empty if an undefined key is pressed. In this case, only the tag (`CMD`) is stacked (or a null line if READ NOTAG is specified).
   - `ETK string`

     where `string` is the ENTER key definition.
   - `FIL n1 n2 n3 string`

where:

n1 n2	are the line number and column number of the beginning of the line on the screen.
n3	is the corresponding file line number. This is returned ONLY if the READ subcommand was issued with the NUMBER option, otherwise it is omitted.
string	is the changed file line. (String can be empty if the ERASE EOF key was pressed.)

- `PAK n string`

  where:

n	is the number of the PA key pressed to terminate the READ.
string	is the PA key definition.

  PAK lines are stacked LIFO.

- `PFK n string`

  where:

n	is the number of the PF key pressed to terminate the READ.
string	is the PF key definition.

  PFK lines are stacked LIFO.

- `PRF n1 n2 n3 string`

  where:

n1 n2	are the line number and column number of the prefix area on the screen.
n3	is the corresponding file line number associated with the prefix area. The file line number is returned only if the READ subcommand was issued with the NUMBER option, otherwise it is omitted.
string	is whatever was typed in the prefix area. (The `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. (The `string` can be empty if the ERASE EOF key was pressed.)

4. If a command is entered on a screen in `MORE` or `HOLDING` status, the command is placed in the CMS input queue (which is the same as the terminal input buffer). For more information on the program stack and terminal input buffer, see *z/VM: REXX/VM User's Guide*.
5. When a READ subcommand is issued from a REXX application, use both the QUEUED() and EXTERNALS() functions to determine if there is anything in the program stack or the terminal input buffer. For information on the QUEUED and EXTERNALS functions, see *z/VM: REXX/VM Reference*.

Responses

The message `Macro-read` *n* `File(s)` is displayed in the status area.

Messages and Return Codes

509E *Subcommand* **subcommand not valid from a prefix macro [RC=4]**
520E Invalid operand: *operand* **[RC=5]**

where return codes are:

0	Normal
1	TOF or EOF reached
4	Invalid when issued from a prefix macro
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Console stack not empty

RECOVER

```
        ┌─1─┐
►►──RECover─┼───┼────────────────────────────────────────────────────────────►◄
        ├─n─┤
        └─*─┘
```

Purpose

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

Operands

n is the number of lines removed by a DELETE, a MERGE, or a PUTD subcommand, or a D prefix subcommand you wish to replace in the file. If *n* is not specified, only the last line that was removed is reinserted in the file.

* all deleted lines are replaced.

Usage Notes

1. The last lines that were deleted are the first to be recovered. For instance, a DELETE 10 followed by a RECOVER 5 would recover the *last* five lines of the 10 deleted lines.
2. Once a deleted line is recovered, it is removed from the buffer that contains recoverable lines. Therefore, DELETE followed by multiple RECOVER subcommands cannot be used to duplicate a line at various locations in a file.
3. When multiple files are being edited, there is only *one* buffer for all removed lines. Thus, a line could be deleted in one file and recovered in another file if the width of the two files is the same.
4. RECOVER subcommands do *not* have to match DELETE subcommands. The following is a valid sequence:
   ```
   delete 2
     .
     .
     .
   delete 3
     .
     .
     .
   recover 1
     .
     .
     .
   recover 4
   ```
5. The size of the recoverable lines buffer depends on the amount of virtual storage.
6. Macros, such as JOIN, that use the DELETE, MERGE or PUTD subcommands put lines in the buffer that contains recoverable lines.

7. If you are editing a file in update mode (where the status area indicates `Update-mode` *n* `File(s)`), deleted lines reflected in an update file cannot be recovered. See Chapter 2, "The XEDIT Command," on page 11 for more information on update mode.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Responses

The recovered line(s) is inserted at the current line. The following message is displayed:

```
502I  nn  line(s)  recovered
```

The recovered line becomes the new current line.

Messages and Return Codes

502I {No|*nn*} line(s) recovered [RC=0 or 3]
520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**

where return codes are:

0	*n* lines have been inserted
3	Pool of deleted lines is empty
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

REFRESH

```
►►──REFRESH──────────────────────────────────────────────────────────────►◄
```

Purpose

Use the REFRESH subcommand to display the screen. Issued from a macro, it presents the screen as of that moment in processing, without waiting for input.

Notes for Macro Writers

1. REFRESH can be used to update the display on the screen without exiting the macro.

Messages and Return Codes

117S Error writing to display terminal [RC=8]
520E Invalid operand: *operand* **[RC=5]**
529E Subcommand is only valid in {display|editing} mode [RC=3]

where return codes are:

0	Normal
3	Subcommand valid only for display terminal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	I/O error writing to screen

RENUM

```
           ┌─10──startno──────────────┐
►►──RENum──┼──────────────────────────┴───────────────────────────────────────►◄
           │                ┌─startno─┐
           └──startno───────┤         ├─┘
                            └─incrno──┘
```

Purpose

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

Operands

startno
: specifies the first line number of the file. If *startno* is not specified, the file starts with line number 10.

incrno
: specifies the value by which each line number is incremented. If *incrno* is not specified, the value specified as *startno* is the default.

Usage Notes

1. The *startno* and *incrno* values cannot exceed 99999 for VSBASIC files or 99999999 for FREEFORT files.
2. This subcommand is intended to be used in EDIT migration mode; XEDIT does not perform line number editing.
3. This subcommand should not be used for a VSBASIC file that has a logical record length greater than 256.
4. See “SET SERIAL” on page 341 for information on controlling file serialization.

Responses

If you are not in a VSBASIC or FREEFORT file and you enter the RENUM command, the following message is displayed:

```
Wrong file format for RENUM
```

When RENUM finds an incorrect VSBASIC or FREEFORT statement in your file, one of the following messages is displayed:

```
Maximum line number exceeded
Overflow at statement number
Invalid syntax in statement number
Invalid line number reference in statement number
Line number referenced in statement number not found
```

Messages and Return Codes

002E File *fn ft fm* not found [RC=28]
037E Filemode *mode* is accessed as read/only [RC=12]
069E Filemode *mode* not accessed [RC=36]
104S Error *nn* reading file *fn ft fm* from disk or directory [RC=31, 55, 70, 99, or 100]

105S	**Error *nn* writing file *fn ft fm* on disk or directory [RC=31, 55, 70, 99, or 100]**
520E	**Invalid operand: *operand* [RC=5]**
531E	**Disk or file space is full; set new filemode or clear some space [RC=13]**
535E	**Invalid parameters for RENUM [RC=5]**
543E	**Invalid number: *number* [RC=5]**
554E	**Not enough virtual storage available [RC=104]**

where return codes are:

0	Normal
5	Invalid operand or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	File mode is accessed as read/only
13	Minidisk or file space is full
28	File not found
31	A rollback occurred
32	Format or content of input file invalid
36	Mode not accessed
55	APPC/VM communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
99	A required system resource is not available
100	Error reading/writing file to disk or directory
104	Out of virtual storage

REPEAT

```
          ┌─1──────┐
►►──REPEat─┼────────┼──────────────────────────────────────────────────────►◄
          ├─target─┤
          └─*──────┘
```

Purpose

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.

Operands

target
: specifies the number of times the REPEAT subcommand executes, starting on the line following the current line. REPEAT computes the number of lines between the current line and the line preceding the target line. The last subcommand entered is then repeated for this number of times. If *target* is not specified, the previous subcommand is executed on the line following the current one.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see “LOCATE” on page 147 and *z/VM: XEDIT User’s Guide*.

*
: the previous subcommand is repeated up to the end of file.

Usage Notes

1. If the *target* is in a forward direction, REPEAT is equivalent to:
   ```
   next 1
   =
   ```
2. If the *target* is in a backward direction, that is, specified with a leading minus (–) sign, REPEAT is equivalent to:
   ```
   up 1
   =
   ```
3. If the previous subcommand contains a *target* or is a subcommand whose purpose is to move the line pointer (such as NEXT), the execution of the REPEAT subcommand may occur beyond the line specified by the *target* of the REPEAT.
4. If the previous subcommand results in a nonzero return code, the execution of the REPEAT subcommand terminates on that line.
5. When repeating a subcommand that resets the line pointer be careful to specify a target line that will actually be reached—otherwise, the target will never be reached and the subcommand will continuously execute.

   For example, the following will execute continuously:
   ```
   :1 input anyline to repeat
   ```

Responses

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

Messages and Return Codes

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

where return codes are:

1	TOF or EOF reached
2	Target not found
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
nn	Same as the repeated subcommand's return codes

REPLACE

```
               (1)
►►──Replace──┬──────┬────────────────────────────────────────────────────►◄
             └─text─┘
```

Notes:

1 If *text* is not specified, the editor deletes the current line and enters input mode.

Purpose

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

Operands

text
: specifies an input line to replace the current line. If *text* is specified, the editor puts it into the file in place of the current line. If *text* is not specified, the editor deletes the current line and enters input mode.

Usage Notes

1. If you issue a REPLACE subcommand when the current line is the Top of File line, the text is inserted after the Top of File line. If you issue it when the current line is the End of File line, the text is inserted before the End of File line.
2. If SET SPILL OFF is in effect (the default), characters that have been pushed beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines starting with the first character or word that would have gone beyond the truncation column.
3. If REPLACE is issued without operands, the current line is deleted and input mode is entered. If you do not enter any data, the deleted line is recovered. However, if the editor is in update mode the deleted line is not recovered.
4. If SET IMAGE ON is in effect, tabs are converted to blanks before a line is inserted into the file and the text inserted will start in the first tab column.

Examples

In the following example, the REPLACE subcommand is used to replace the current line with specified text.

Current Line

```
===== This is the current line.

replace Now this is the new current line.

===== Now this is the new current line.
```

Responses

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

```
573I Input mode:
```

Messages and Return Codes

503E {Truncated|Spilled} [RC=3]
557S No more storage to insert lines [RC=4]

where return codes are:

0	Normal
3	Truncated or spilled
4	Not enough storage to add lines
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

RESET

```
►►──RESet──────────────────────────────────────────────────────────────►◄
```

Purpose

Use the RESET subcommand to remove all prefix subcommands or macros when the screen is in a *pending* status.

Usage Notes

1. The RESET subcommand removes all prefix subcommands and macros when the screen displays a pending status.
2. To remove prefix subcommands or macros when the screen is not in a pending status, press CLEAR.
3. RESET does not reset the prefix area of shadow lines or any lines outside the defined scope. See “SET SCOPE” on page 327.

Notes for Macro Writers

1. If a user enters RESET when a prefix macro is pending, that macro is invoked with the argument string PREFIX CLEAR *pline* where *pline* is the line number for the line on which the prefix macro was pending. When the macro examines the argument string and finds it was invoked with the CLEAR argument, it would (usually) stop processing. However, the prefix macro may issue subcommands to change the current line and specify which line is current when it is finished executing and the screen is redisplayed. (For more information on the argument string automatically passed to a prefix macro, see *z/VM: XEDIT User’s Guide*.)

Examples

If you initiate copying or moving a block of lines by typing MM or CC in the first and last lines of the block, but you have not yet specified the destination line (with P or F), you can cancel the operation with RESET.

Responses

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

Messages and Return Codes

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

where return codes are:

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

RESTORE

```
►►──RESTore──────────────────────────────────────────────────────────►◄
```

Purpose

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

Usage Notes

1. See "PRESERVE" on page 180 for a list of variables the RESTORE subcommand affects.
2. If the column pointer was located outside the restored zone settings, it is repositioned to the left or right zone. If the column pointer was positioned to the left of the new zone, it moves to Top of Line (TOL). If it was positioned beyond the new right zone, it moves to End of Line (EOL).

Messages and Return Codes

507E No preserved data to restore [RC=3]
520E Invalid operand: *operand* **[RC=5]**

where return codes are:

0	Normal
1	The column pointer was located outside the restored zone settings
3	No PRESERVE has been issued
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

RGTLEFT (Macro)

```
                 (1)
►►──RGTLEFT──┬─────┬──────────────────────────────────────────────►◄
             └─n───┘
```

Notes:

1 If *n* is not specified, the screen is moved up to a maximum of three-fourths of the logical screen width.

Purpose

Use the RGTLEFT macro to view columns of data not currently visible on the screen.

Operands

n is the number of columns to move the screen. The VERSHIFT setting determines whether the screen moves right or left (see Usage Note 3). If *n* is not specified, the screen is moved up to a maximum of three-fourths of the logical screen width (for example, 60 characters on an 80-character screen). If the logical record length is less than this value (screen width plus 3/4 screen width), the screen moves only enough to display the rest of the logical record, up to a maximum of three-fourths of the first verify width (the number of columns in the first verify pair). For example, if VERIFY is set at 1 40 on a file with a record length of 80, the first RGTLEFT command moves the screen right 30 columns (3/4 of 40) and columns 31 through 70 are displayed.

Usage Notes

1. The RGTLEFT macro does not cause data to be lost, nor does it move the line or column pointer.
2. The RGTLEFT macro is most useful when assigned to a PF key. The PF10 key is initially set to RGTLEFT. Pressing PF enables you to see data to the right of the screen; pressing PF again returns the screen to the original view of the file.
3. If VERSHIFT is less than or equal to zero, the view is moved to the right, making the VERSHIFT value greater than zero. When VERSHIFT is greater than zero, the view is moved back to the left, returning to the original display of the file.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
529E	**RGTLEFT is only valid in {display|editing} mode [RC=3]**
543E	**Invalid number:** *number* **[RC=5]**
576E	**{Total verify width exceeds screen size (***nn***)|Total offset exceeds LRECL (***nn***)} [RC=5]**

where return codes are:

0	Normal
3	RGTLEFT valid in display mode only
5	Invalid operand or number

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

RIGHT

```
            ┌─1─┐
►►──RIght───┼───┼──────────────────────────────────────────────────────────►◄
            └─n─┘
```

Purpose

Use the RIGHT subcommand to view columns of data 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.

Operands

n specifies the number of positions to the right of the last column on the screen you want to see. If *n* is not specified, one position to the right of the last column becomes visible.

Usage Notes

1. The RIGHT subcommand does not cause data to be lost, nor does it move the line or column pointer.
2. To get the data back to its original position, use the LEFT *n* subcommand. See "RGTLEFT (Macro)" on page 219.
3. RIGHT subcommands are cumulative. For example entering the subcommands:

   ```
   ri 10
   ri 10
   ```

   is equivalent to entering:

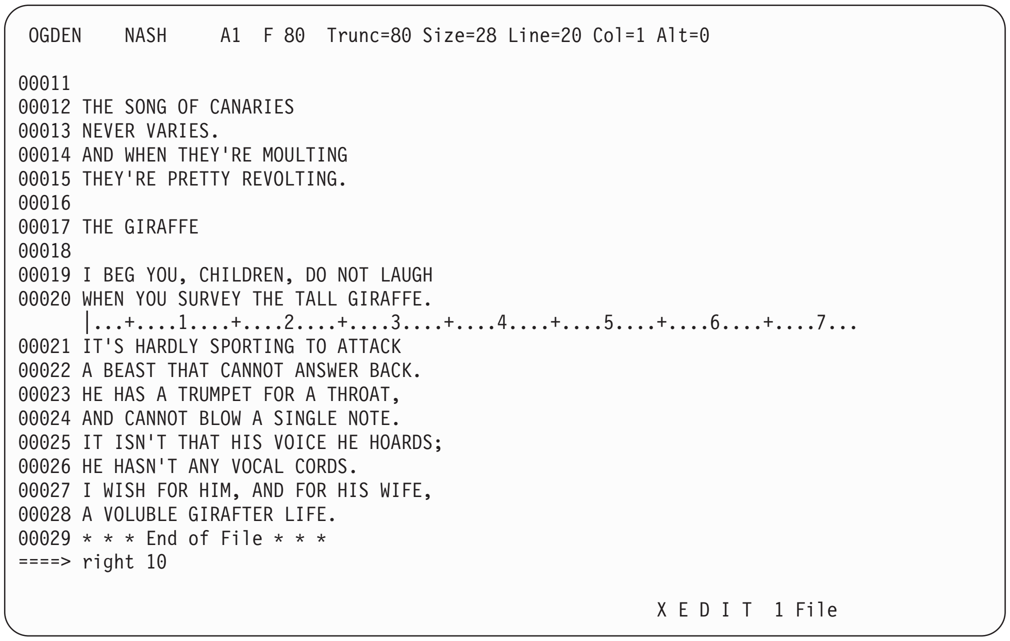
   ```
   ri 20
   ```
4. If you have issued several RIGHT or LEFT subcommands and have forgotten the total value of *n*:
   a. RIGHT 0 or LEFT 0 restores the screen to its original display.
   b. SET VERIFY resets RIGHT (or LEFT) to zero.
   c. QUERY VERSHIFT displays *n* (the result of a RIGHT) or –*n* (the result of a LEFT).
5. The total number of columns moved cannot exceed the logical record length.

Notes for Macro Writers

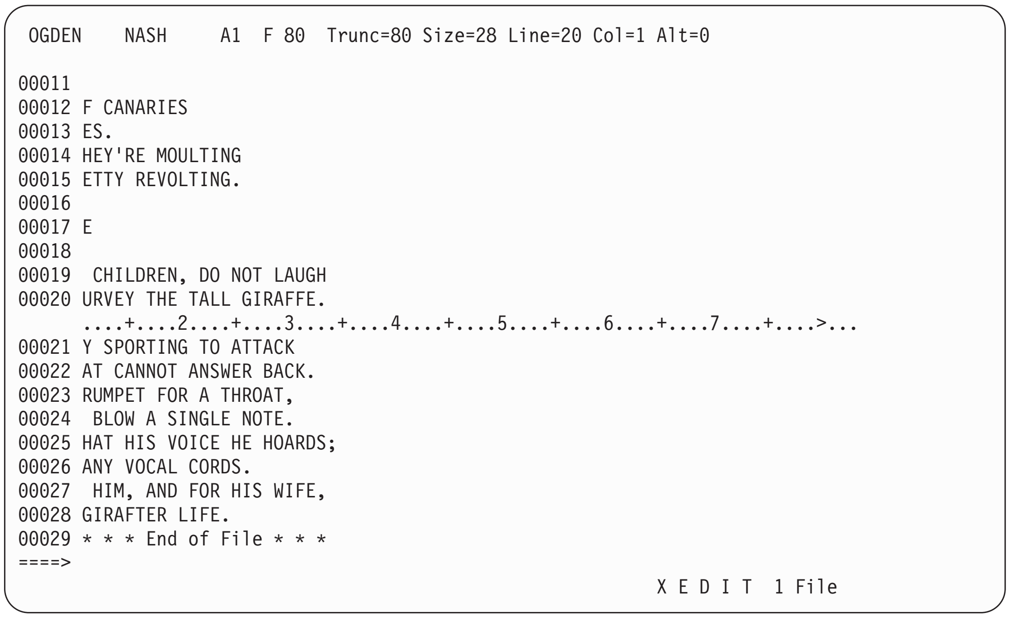
EXTRACT/VERSHIFT/ returns the value of *n* or –*n*.

Examples

Figure 12 on page 222 is a before-and-after example of the RIGHT subcommand. Line 20 is the current line.



```
 OGDEN    NASH     A1  F 80  Trunc=80 Size=28 Line=20 Col=1 Alt=0

00011
00012 THE SONG OF CANARIES
00013 NEVER VARIES.
00014 AND WHEN THEY'RE MOULTING
00015 THEY'RE PRETTY REVOLTING.
00016
00017 THE GIRAFFE
00018
00019 I BEG YOU, CHILDREN, DO NOT LAUGH
00020 WHEN YOU SURVEY THE TALL GIRAFFE.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00021 IT'S HARDLY SPORTING TO ATTACK
00022 A BEAST THAT CANNOT ANSWER BACK.
00023 HE HAS A TRUMPET FOR A THROAT,
00024 AND CANNOT BLOW A SINGLE NOTE.
00025 IT ISN'T THAT HIS VOICE HE HOARDS;
00026 HE HASN'T ANY VOCAL CORDS.
00027 I WISH FOR HIM, AND FOR HIS WIFE,
00028 A VOLUBLE GIRAFTER LIFE.
00029 * * * End of File * * *
====> right 10

                                                            X E D I T  1 File
```



```
 OGDEN    NASH     A1  F 80  Trunc=80 Size=28 Line=20 Col=1 Alt=0

00011
00012 F CANARIES
00013 ES.
00014 HEY'RE MOULTING
00015 ETTY REVOLTING.
00016
00017 E
00018
00019  CHILDREN, DO NOT LAUGH
00020 URVEY THE TALL GIRAFFE.
      ....+....2....+....3....+....4....+....5....+....6....+....7....+....>...
00021 Y SPORTING TO ATTACK
00022 AT CANNOT ANSWER BACK.
00023 RUMPET FOR A THROAT,
00024  BLOW A SINGLE NOTE.
00025 HAT HIS VOICE HE HOARDS;
00026 ANY VOCAL CORDS.
00027  HIM, AND FOR HIS WIFE,
00028 GIRAFTER LIFE.
00029 * * * End of File * * *
====>
                                                            X E D I T  1 File
```

Figure 12. RIGHT Subcommand — Before and After

Responses

The screen moves to the right relative to the file data.

Messages and Return Codes

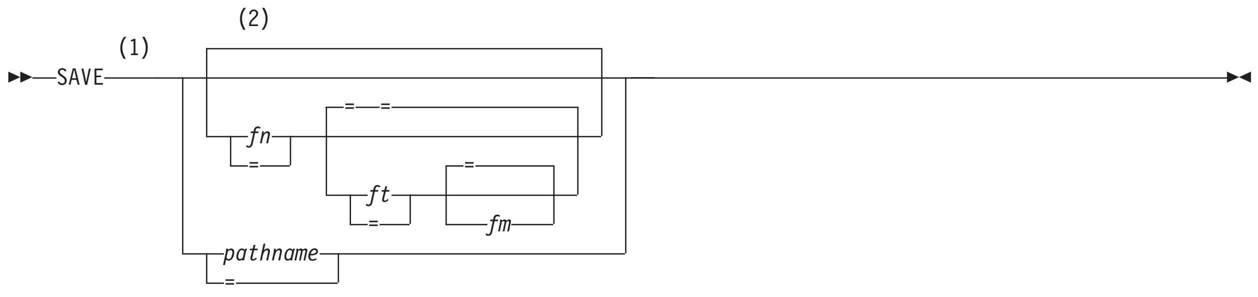
520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
576E {Total verify width exceeds screen size (*nn***)|Total offset exceeds LRECL (***nn***)} [RC=5]**

where return codes are:

0 Normal

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

SAVE



Notes:

1 The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the byte file system (BFS) format required for BFS files (*pathname*).

2 If file ID is not specified, the default file ID is = = = (for NAMETYPE CMS) or = (for NAMETYPE BFS).

Purpose

Use the SAVE subcommand or SSAVE to write the edited file on disk, into a Shared File System (SFS) directory, or into a byte file system (BFS) regular file without returning control to the environment that invoked the editor. You may also use the SAVE subcommand to change the file identifier.

Operands

fn indicates the file name of the SFS or minidisk file to be saved (or member name when editing a member of a MACLIB). If you specify only *fn*, the file type and file mode remain the same.

ft indicates the file type of the SFS or minidisk file to be saved.

fm indicates the file mode of the file to be saved.

pathname
is the name of the BFS file to be saved. See “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the different forms of the BFS path name.

The length of the path name entered on this subcommand is limited. Use `SET PNAME` *pathname_fragment1* (or `PN` *pathname_fragment1*) and `SET PNAME APPEND` *pathname_fragment2* one or more times, and then use `SAVE =` if you wish to use a longer path name. Refer to Chapter 1, “Rules and Conventions,” on page 1 for more information about the maximum lengths allowed for XEDIT subcommands.

Usage Notes

1. To use the SAVE subcommand to write a file to a minidisk, you must have the minidisk accessed in read/write mode.
2. To use the SAVE subcommand to *change* another user’s SFS file, you must be properly authorized:

- For a file in a directory control directory, you must have directory control write authority to the directory, and you must access the directory in read/write mode.

  When you access another user's directory, it is, by default, accessed in read-only mode. To access another user's directory in read/write mode, specify the FORCERW option on the ACCESS command.
- For a file in a file control directory, you must have write authority to the file; the directory can be accessed in either read/write or read-only mode. However, if the directory is accessed in read-only mode and the CMS command SET RORESPECT ON has been issued, an attempt to SAVE will fail.

3. To use the SAVE subcommand to *create* a file in another user's SFS directory, you must be properly authorized:
   - For directory control directories, you need directory control write authority to the directory, and you must access the directory in read/write mode. When the file is created, the owner of the directory becomes the owner of the new file, but because you have directory control write authority, you will be able to write to the file.

     When you access another user's directory, it is, by default, accessed in read-only mode. To access another user's directory in read/write mode, specify the FORCERW option on the ACCESS command.
   - For file control directories, you need write authority to the user's directory. In this case, the directory can be accessed in either read/write or read-only mode. However, if the directory is accessed in read-only mode and the CMS command SET RORESPECT ON has been issued, an attempt to SAVE will fail. When the file is created, the owner of the directory becomes the owner of the new file, but because you created the file, you have write authority for the file.
4. To use the SAVE subcommand to change a BFS file, you must have permission to write to the file. To create a file in a BFS, you must have permission to write to the parent directory.
5. Permissions for a new BFS file are set based on the current value of the mask, with the exception of execute permissions, which are not set on. Use OPENVM PERMIT to change the defaults. See *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM for more information on these OPENVM commands.
6. The BFSLINE value determines whether end-of-line characters are inserted in the file. The RECFM value determines whether trailing blanks are stripped from the file. Refer to "SET BFSLINE" on page 245 and "SET RECFM" on page 319 for more information.
7. For SFS and minidisk files, use the QUERY DIRATTR command or the QUERY ACCESSED command to determine whether the directory has the FILECONTROL or the DIRCONTROL attribute. (In the QUERY ACCESSED display, the *Vdev* column contains *DIR* for FILECONTROL directories and *DIRC* for DIRCONTROL directories.)
8. If you specify a new file identifier, the original copy of the file on disk or in the directory or the BFS is not changed. Changing the file identifier simply creates another copy of the file with a new file identifier.
9. When NAMETYPE CMS is in effect, specifying an equal sign (=) for the *fn, ft,* and *fm* operands means the corresponding FNAME, FTYPE, and FMODE values are used. If not specified as an equal sign, the operand will be translated to uppercase.

When NAMETYPE BFS is in effect, specifying an equal sign (=) for the *pathname* operand means the value set using SET PNAME will be used. The path name is converted to uppercase only when SET CASE UPPER is in effect.

If you change the NAMETYPE value, and then attempt to use an equal sign without setting the appropriate FNAME, FTYPE, or PNAME values, the editor returns an error. If you change the NAMETYPE value to CMS and then use an equal sign without setting FMODE, it defaults to file mode A1.

10. SAVE is defined as a synonym to PSAVE (the protected equivalent of SAVE). The PSAVE subcommand issues the

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

    message if:

    - the *disk* file was changed outside of this editing session. For example, if you were editing the same file in two different rings, changed the file in the second ring, filed it, changed the file in the first ring and attempted to file that one, the 594E message would be displayed.

    If you are sure you want to save the file you are editing, issue SSAVE (unprotected SAVE, abbreviated SS) to save the file and overlay any existing version. SSAVE is a synonym for COMMAND SAVE.
11. If you try to SAVE a shared file or BFS file another user has modified during your editing session, the editor stops the SAVE operation and displays message 594E.

    ```
    594E File {fn ft fm|pathname} already exists or changed;
         use FFILE or SSAVE
    ```

    At this point you can decide whether to write over the other user's changes or preserve them. If you want to write over them, deleting the changes, enter SSAVE. If you want to preserve the other user's changes, reissue the SAVE subcommand using a unique file identifier to save the changes under a different name. Later, you can determine the differences between the two versions of the file and combine them. Note that the LOCK option of the XEDIT command prevents other users from changing an SFS file control while you are editing it.
12. If you change the member name (to a unique member name) and the member has previously been written to disk or directory, that copy with the original name of the member is not altered.

    However, if you change the member name while editing a member so the new member name is identical to that of an existing member, the editor stops the SAVE operation and displays the following warning:

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

    If you want to write over the existing member, enter SSAVE (abbreviated as SS). If not, change the member name so it is unique and reissue the SAVE subcommand.
13. Members with identical names contained in libraries with identical names, but residing on different disks or directories, are not written over if you attempt to change the file mode. The following warning is issued:

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

    For example:

```
PROJA MACLIB A1             PROJA MACLIB B1
ENTER                       ENTER
LEAVE                       LEAVE
FORMAT                      FORMAT
COMPUTE                     COMPUTE
```

If you are editing the ENTER member in the PROJA MACLIB that resides on your file mode B and attempt to change the file mode by issuing `SAVE = = A`, the warning message is issued.

14. If you try to SAVE an empty file, you will get the following message:

    ```
    595E Issue SSAVE/FFILE [to a directory] to write an empty file
         or QQUIT to exit without writing file
    ```

    SSAVE/FFILE will only succeed for empty files which reside in an SFS or BFS directory. In addition, since at least one record is required for a packed file, empty packed files are not supported. Empty autosave files are not written. Empty MACLIB members are also unsupported. When in update mode, any update that deletes all the records in the merged source file is not allowed.

    For any unsupported case, you will receive the following error message:

    ```
    559E Empty file fn ft fm not written
    ```

15. Members of a MACLIB must have a file type of MEMBER.
16. When editing a file in update mode, the file written will contain changes to the file defined by the latest update level applied plus any changes you made while editing. The original source file to which the update(s) were applied is *not* changed. If the updates were merged (see the MERGE option on the XEDIT command), the file written will contain changes defined by all update levels plus any changes you made.

    If the latest update level contained update control comment records, they will be placed at the beginning of the file written, regardless of their original position in the update file. See the CMS UPDATE command in *z/VM: CMS Commands and Utilities Reference* for more information on the contents of an update file.
17. If you want a file to be saved automatically at regular intervals, use the SET AUTOSAVE subcommand. AUTOSAVE writes to an SFS or minidisk file, even when the NAMETYPE setting is BFS.
18. If the automatic save function is in effect (SET AUTOSAVE subcommand), the corresponding AUTOSAVE file is erased when a SAVE is executed.
19. To write a file to disk, directory, or to the BFS and end the editing session, use the FILE subcommand instead of the SAVE subcommand.
20. If the SAVE subcommand is used to create a file in an SFS file control directory, you can use the CMS CREATE LOCK command to prevent other users from modifying the file while you are editing it.
21. When you XEDIT a BFS file and attempt to write the file when the record format is V (variable), you will see an error message if the line length exceeds 65 535. (The maximum record length for a variable file in the CMS record file system is 65 535.)
22. The editor will use the current NAMETYPE value to interpret whether the file ID is in the CMS format required for SFS and minidisk files (*fn ft fm*) or in the BFS format required for BFS files (*pathname*). NAMETYPE may be set using an XEDIT option, or by the SET NAMETYPE subcommand.
23. Commit behavior of BFS files is different from that of SFS or minidisk files. For example, if the editor encounters an error when writing out a minidisk or SFS file, your changes are reversed and the original file is preserved. This is not

the case for BFS files. The editor creates a copy of the file in XEDTEMP CMSUT1 on your A disk so you can recover your data if an error occurs when writing the file to the byte file system.

If the editor is unable to create the XEDTEMP CMSUT1 file, you will receive a message describing the problem, and a second message:

```
595E  Not able to create CMS file used for recovery.  Correct
      error or QQUIT to exit without writing file
```

If the editor encounters an error while writing the byte file system file such that the contents of an existing file are damaged, you will receive a message describing the problem, and a second message to tell you that you must take action to recover your file.

```
024E  File XEDTEMP CMSUT1 A1 contains file contents; use
      OPENVM PUT to recover file
```

24. When a new BFS file is created, the owning UID established is the effective UID of the VM User ID on which the request was issued. The GID is the GID of the parent directory. Use OPENVM OWNER to change the defaults. Refer to *z/VM: OpenExtensions Commands Reference* or enter HELP OPENVM OWNER for more information.

Messages and Return Codes

002E File *fn ft fm* **not found [RC=28]**
007E File *fn ft fm* **is not fixed, 80-character records [RC=32]**
024E File XEDTEMP CMSUT1 A1 contains file contents; use OPENVM PUT to recover file
033E File [*fn ft fm***] is not a {library|regular BFS file} [RC=32]**
037E Base file for *fn ft fm* **is in a DIRCONTROL directory accessed read/only [RC=12]**
037E Filemode *mode* **is accessed as read/only [RC=12]**
039E No entries in library *fn ft fm* **[RC=32]**
048E Invalid filemode *mode* **[RC=24]**
054E Incomplete [or incorrect] fileid specified [RC=24]
062E Invalid character in fileid {*fn ft fm***|***pathname***} [RC=20]**
069E Filemode *mode* **not accessed [RC=36]**
104S Error *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, 99, or 100]**
105S Error *nn* **writing file** *fn ft fm* **on disk or directory [RC=55, 70, 76, 99, or 100]**
157S MACLIB limit exceeded [RC=88]
167S Previous MACLIB function not finished [RC=88]
229E Unsupported OS dataset, error *nn* **[RC=80, 81, 82, 83, or 84]**
512E This is not allowed in CMS subset mode [RC=100]
520E Invalid operand: *operand* **[RC=5]**
531E BFS file space is full; clear some space [RC=13]
531E Disk or file space is full; set new filemode or clear some space [RC=13]
532E Disk or file space is full; AUTOSAVE failed
554E Not enough virtual storage available [RC=104]
559E Empty file *fn ft fm* **not written [RC=88]**
560E Not enough space for serialization between TRUNC and LRECL
579E Records truncated to *nn* **when added to** *fn ft fm* **[RC=3]**
594E File {*fn ft fm***|***pathname***} already exists or changed; use FFILE or SSAVE [RC=3]**
595E Issue SSAVE/FFILE [to a directory] to write an empty file or QQUIT to exit without writing file [88]

595E	**Not able to create CMS file used for recovery. Correct error or QQUIT to exit without writing file**
595E	**Not able to obtain lock. Issue SSAVE/FFILE to write file without locking or QQUIT to exit without writing file [RC=70]**
598S	**Unable to build update file: internal list destroyed [RC=7]**
599S	**Unable to build update file: serialization destroyed [RC=7]**
622E	**Insufficient free storage for reading map [RC=104]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code** *rc* **and reason code** *rs* **[RC=55, 104]**
1138E	**File sharing conflict for file {***fn ft fm*|*pathname***} [RC=70]**
1141W	**User filespace threshold exceeded**
1184E	**A directory is not found, or you are not permitted to use a directory in** *pathname* **[RC=28]**
1215E	**File** *fn ft fm* **is locked or in use by another user [RC=70]**
1258E	**Not {authorized|permitted} to write file {***fn ft fm*|*pathname***} [RC=12]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=55, 70, 76, 99, or 100]**
1301S	**Rollback error** *nn***, file** *fn ft fm* **left open**
1350E	**AUTOSAVE must be targeted to your own directory [RC=12]**
2105E	**Permission is denied [24]**
2120E	**Unable to resolve path name** *pathname* **[RC=24]**
2131E	**[FNAME] [FTYPE] [PNAME] {is|are} not set and NAMETYPE {CMS|BFS} is in effect [RC=24]**
2134E	**Return code** *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname***] [RC=100]**
2154E	**File {***fn ft fm*|*pathname* **is migrated and implicit RECALL is set to OFF [RC=50]**
2155E	**DFSMS/VM error occurred during creation or recall of file {***fn ft fm*|*pathname***} [RC=51]**
2526E	**File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code** *acs rcode* **[RC=51]**

where return codes are:

0	Normal
3	Truncated or spilled, or file already exists
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
7	Error building the update file
12	Minidisk defined in file mode is read-only; or not authorized to write to file; or cannot AUTOSAVE to another user's directory or you do not have permission for the BFS file or byte file system mounted read-only
13	Minidisk or file space or byte file system is full
20	Character in file name or file type or path name is not valid
24	File mode or path name is not valid
28	File not found
31	A rollback occurred
32	File is not a library, or library has no entries, or file is not fixed, 80 char. records or BFS file is not a regular file, or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error

70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
84	Unsupported OS data set
88	Previous MACLIB function not finished, MACLIB limit exceeded, or unsupported attempt to write an empty file
99	A required system resource is not available
100	Error *nn* {reading|writing} file {from|to} disk or directory or byte file system
104	Insufficient storage available

SCHANGE (Macro)

```
            ┌─6─────────┐
►►──SCHANGE─┼───────────┼──────────────────────────────────────────────────────►◄
            └─keynumber─┘
```

Purpose

Use the SCHANGE (selective change) macro to locate every occurrence of a string and to change that string only when you choose.

The SCHANGE macro can be used when it has been assigned to a PF key (with the SET PF*n* subcommand). You can also type a CLOCATE or CHANGE subcommand on the command line before pressing the PF key assigned to the macro SCHANGE. By default, the SCHANGE macro automatically repeats the CLOCATE or CHANGE subcommand saved in the LASTLORC buffer (see "SET LASTLORC" on page 281).

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 you press the PF key assigned to SCHANGE (with the SET PF*n* subcommand), 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 you have typed a CHANGE /*string1*/*string2*/ subcommand in the command line, pressing the *second* PF key changes *string1* to *string2*. (This second PF key has no effect when the CLOCATE subcommand is used.)

For an example of how to use the SCHANGE macro when it has been assigned to a PF key, see *z/VM: XEDIT User's Guide*.

Operands

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 the editor assigns, but any number can be used.

Usage Notes

1. Pressing the PF key assigned to SCHANGE causes the cursor to move but does not cause the line pointer to move (unless the screen is scrolled forward). The column pointer moves to the column where the specified string begins.
2. The screen scrolls forward automatically after all occurrences of the string have been located on the current screen.

   The screen does not scroll forward if the string does not appear in the rest of the file.
3. Pressing the PF key assigned to SCHANGE places the cursor under the first character of the string.

   Pressing the second PF key replaces *string1* with *string2*.

4. The advantage of using SCHANGE with CLOCATE (rather than CLOCATE and =) is the cursor is placed under the matching string.
5. SCHANGE does not change a string located to the left or right of the screen. In order for SCHANGE to locate all string targets, you must change SET VERIFY to 1 and *.
6. SCHANGE can be assigned to a PA key instead of a PF key; however, the key number specified on the SCHANGE subcommand must be that of a PF key.
7. When AUTOSAVE is set ON, the file is not autosaved until SCHANGE completes execution. Entering a subcommand other than the two PF keys assigned to SCHANGE and CHANGE completes execution of the SCHANGE macro.
8. When using SCHANGE with the CHANGE subcommand, the delimiters *, −, +, :, ¬, and the digits 0 through 9 are not valid. Use another character as the delimiter for the CHANGE subcommand.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Responses

Each time a string is located, the cursor is placed under the first character of the string, and the line containing the string is highlighted.

If a CLOCATE subcommand is typed in the command line or saved in the LASTLORC buffer and you press the first PF key, the following message is displayed when a string is located:

```
Target string1 found
```

If a CHANGE subcommand is typed in the command line or saved in the LASTLORC buffer and you press the first PF key, the following message is displayed when a string is located:

```
String string1 found; --- PFnn set for selective CHANGE
```

When you press the second PF key, the following message is displayed:

```
String string1 changed to string2
```

If you press the PF key assigned to SCHANGE without first typing a CHANGE or CLOCATE in the command line, and the LASTLORC buffer does not contain a CHANGE or CLOCATE subcommand, the following message is displayed:

```
No CHANGE or CLOCATE subcommand specified
```

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
525E	**Invalid {PFkey|PFkey/PAkey} number [RC=5]**
529E	**Subcommand is only valid in {display|editing} mode [RC=3]**
545E	**Missing operand(s) [RC=5]**
561E	**Cursor is not on a valid data field [RC=1 or 3]**
569E	**No CHANGE or CLOCATE subcommand specified [RC=5]**
574E	**CHANGE not valid {with CLOCATE|after cursor movement}**
586E	**Not found [RC=2]**

where return codes are:

0	Normal
2	Not found

3	Invalid placement of cursor, or subcommand is valid only for display terminal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro caller from the last file in the ring

SET

```
►►──SET──options──────────────────────────────────────────────────────►◄
```

Purpose

Use the SET subcommand to change the settings of various editing options while editing is in progress. You can specify only one option in each SET subcommand and cannot issue SET without an option. Use the QUERY subcommand to display the initial settings of SET options or the values set by previously issued SET subcommands.

Operands

The options available with SET are summarized in the following list. A complete description of each option follows.

ALT	IMPcmscp	SCOPE
APL	LASTLorc	SCReen
ARBchar	LINENd	SELect
AUtosave	LRecl	SERial
BFSLine	MACRO	SHADow
BRKkey	MASK	SIDcode
CASE	MSGLine	SPAN
CMDline	MSGMode	SPILL
COLOR	NAMetype	STAY
COLPtr	NONDisp	STReam
CTLchar	NULls	SYNonym
CURLine	NUMber	TABLine
DISPlay	PACK	TABS
ENTer	PAn	TERMinal
ESCape	PENDing	TEXT
ETARBCH	PFn	TOFEOF
ETMODE	PName	TRANSLat
FILler	Point	TRunc
FMode	PREfix	VARblank
FName	RANge	Verify
FType	RECFm	WRap
FULLread	REMOte	Zone
HEX	RESERved	=
IMage	SCALe	

Usage 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 options you override are changed. All other options retain their initial setting or default value.
3. You can use the SUPERSET subcommand to enter multiple SET options on one subcommand to improve performance. See “SUPERSET” on page 393 for more information.

4. The following SET subcommand options affect only the view of the file in which the subcommand is entered:

ARBCHAR	LINEND	SPAN
AUTOSAVE	MACRO	SPILL
CASE	MASK	STAY
CMDLINE	MSGLINE	STREAM
COLOR	MSGMODE	SYNONYM (ON|OFF)
COLPTR	NULLS	TABLINE
CURLINE	NUMBER	TABS
DISPLAY	PACK	TOFEOF
ESCAPE	PREFIX (ON|NULLS|OFF)	TRUNC
ETARBCH	RANGE	VARBLANK
FILLER	SCALE	VERIFY
HEX	SCOPE	WRAP
IMAGE	SERIAL	ZONE
IMPCMSCP	SHADOW	=
LASTLORC	SIDCODE	

The following options affect all views of the file, but not other files in the ring:

ALT	LRECL	POINT
BFSLINE	NAMETYPE	RECFM
ENTER	PA*n*	RESERVED
FMODE	PENDING	SELECT
FNAME	PF*n*	
FTYPE	PNAME	

The following options affect all files in the ring:

APL	NONDISP	TERMINAL
BRKKEY	PREFIX (SYNONYM)	TEXT
CTLCHAR	REMOTE	TRANSLAT
ETMODE	SCREEN	
FULLREAD	SYNONYM (*newname*)	

SET ALT

```
                    (1)
>>──SET──ALT──n──┬─────┬──────────────────────────────────────────────><
                 └─p───┘
```

Notes:

1 If *p* is not specified, the total alteration count since SAVE remains unchanged.

Purpose

Use the ALT option to change the alteration count, the number of alterations made to the file since the last AUTOSAVE or SAVE. This count is displayed in the file identification line as Alt=*n*.

Operands

n represents the alteration count since the last AUTOSAVE.

p represents the total alteration count since the last SAVE. If you omit *p*, the total alteration count since SAVE remains unchanged.

Initial Setting

Both counts are initially set to zero.

Usage Notes

1. You can reset both counts. Specifying only one operand (SET ALT *n*) resets the first count. Specifying two operands resets both counts.
2. You must specify the subcommand SET with this option (to avoid conflict with the ALTER subcommand).
3. The alteration count represents the number of subcommands that were executed that changed the file and the number of lines modified when typing on the full screen. The alteration count is incremented whenever any action modifies a line, even if the line does not actually change from its original state. For example, this occurs when you overtype a character in a line or if you press ERASE EOF when the cursor is in column one.

   Macros that issue numerous commands that change the file may increase the count by more than one.
4. SET ALT is intended for use from within a macro. It should not be used for interactive editing because it can interfere with the AUTOSAVE function.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

0	Normal
5	Invalid or missing operand(s) or number

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

SET APL

```
►►──┬─────┬──APL──┬─ON──┬──────────────────────────────────────────────►◄
    └─SET─┘       └─OFF─┘
```

Purpose

Use the APL option to inform the editor and CMS if APL characters are to be used.

Operands

ON
specifies APL characters are to be used. You must issue a SET APL ON before using these characters so proper character code conversion takes place.

OFF
specifies no code conversion is to be performed for APL keys.

Initial Setting

Defined by the CMS SET APL setting.

Usage Notes

1. If a terminal is equipped with the APL feature, special APL keys are available. Before using these keys, you must inform the editor so proper character code conversion takes place. There are two ways to inform the editor:
   a. Issue CMS SET APL ON.
   b. Issue the editor SET APL ON subcommand.
2. Because the conversion is costly, it is recommended you issue the XEDIT subcommand SET APL OFF when you stop using the special keys.
3. Issuing SET APL ON while TEXT is ON causes TEXT to be set to OFF. Similarly, issuing SET TEXT ON while APL is ON causes APL to be set to OFF.
4. Changing the APL setting for XEDIT also changes the APL setting for CMS.
5. In order to use the INSERT or DELETE key before the APL symbol for a lower left-hand box corner (X'1E') and preserve its value, use the XEDIT subcommand SET IMAGE OFF.
6. If you are in a GUI environment, SET APL is ignored.

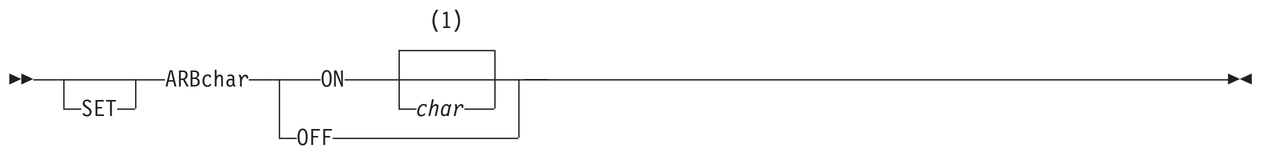
Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET ARBCHAR



Notes:

1 If *char* is not specified, the default is the previous setting.

Purpose

Use the ARBCHAR option to define an arbitrary character for use in a target definition. An arbitrary character between character strings in a target definition means, *ignore all intervening characters.*

Operands

ON
turns on the use of a special character as an arbitrary character.

OFF
turns off 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.

Initial Setting

```
ARBCHAR OFF $
```

Examples

This section shows examples of the SET ARBCHAR subcommand.

Example 1

```
locate  /air$plane/
```

locates in the file:

```
the airplane was landing
```

and also locates in the file:

```
cold air surrounded the plane
```

Example 2

```
clocate  /($)/
```

locates the first expression in the line that is in parentheses.

Example 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 locate in the file:

`airplane` or `cold air surrounded the plane`

Example 4

Using CHANGE with SET ARBCHAR:

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

```
String2 contains more arbitrary characters than string1.
```

For example:

Subcommand:
```
set arbchar on $
```

File Line: `the white house with several large windows`

Subcommand:
```
c /the$house$windows/a$farmhouse$two$shutters/
```

Result:

```
511E    String2 contains more arbitrary characters than string1
```

Subcommand:
```
c /the$house$windows/a$farmhouse$shutters/
```

Result: `a white farmhouse with several large shutters`

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

File Line: `the white house with several large windows`

Subcommand:
```
c/the$house$windows/a$large farmhouse/
```

Result: `a white large farmhouse`

Special use of an arbitrary character:

If a dollar sign ($) is the arbitrary character, it can be used in *string1* or *string2* in the following ways:

string1: `/$A/`

The $ means, *the string starting in the zone1 column and ending with the character that precedes A*. (If no zone is defined, the string starts in column 1.)

Example:

```
change  /$A/A/
```

deletes all characters that precede A within the same zone.

string1: `/B$/`

The $ means, *the string starting with the character following B and ending at the truncation column.* If SET ZONE has been issued, the string ends at the zone2 column rather than the truncation column.

Example:

```
change /B$/B/
```

deletes the characters that follow B.

Messages and Return Codes

511E String2 contains more arbitrary characters than string1 [RC=5]
520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET AUTOSAVE

```
                             ┌─A────┐
►►──┬─────┬──AUtosave──┬──n──┼──────┼─┬──────────────────────────────────────────►◄
    └─SET─┘            │     └─mode─┘ │
                       └─OFF──────────┘
```

Purpose

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.

Operands

n
: is a number that specifies the frequency of the automatic save function. One SAVE subcommand is issued for every *n* subcommands that change, delete, or add lines to the file. In a full screen environment, each line changed (by over-typing) counts as one. The automatic SAVE occurs when this number equals or exceeds *n*.

OFF
: turns off the automatic save function.

mode
: specifies the accessed minidisk or SFS directory on which the file is written. A is the default mode.

Initial Setting

```
AUTOSAVE  OFF
```

Usage Notes

1. QUERY AUTOSAVE displays the current AUTOSAVE setting. The file identification line indicates the number of changes since the last AUTOSAVE (Alt=*n*).
2. When AUTOSAVE is in effect, SAVE or FILE subcommands erase the corresponding AUTOSAVE file. The QUIT subcommand does not erase it.
3. If the system crashes during an editing session, you can recover all changes made up to the time of the last automatic save. To do this, use the CMS COPYFILE command with the REPLACE option to replace the original file with the AUTOSAVE file. Then, erase the AUTOSAVE file and resume editing. COPYFILE keeps any SFS authorities and aliases associated with the original file.

   Another way to recover your changes after a system crash is to erase the original file and rename the AUTOSAVE file to the original file; however, this method may cause you to lose any SFS authorities and aliases associated with the original file.

   Use the OPENVM PUTBFS command to recover changes to a byte file system file. Refer to *z/VM: OpenExtensions Commands Reference* for more information on this command.
4. To recover from an erroneous change to the file, for example, a bad global change, you should immediately issue a QUIT subcommand, replace the

original file with the AUTOSAVE file by using the REPLACE option of the CMS COPYFILE command, erase the autosave file, and continue. However, if the ALT count in the file identification line is 0, the erroneous change has already been saved and this technique does not recover from it.

5. While an XEDIT macro is executing, the automatic save is not performed. The autosave is done when the macro completes execution.
6. If you specify a mode that is either a read-only disk, or is not an accessed disk, the autosave setting becomes ON, and the filemode becomes the mode you specified. So you do not receive any subsequent messages, either access the disk as R/W if possible, or change the mode to a disk that is already accessed as READ/WRITE.
7. If the XEDIT work file, XEDTEMP CMSUT1, exists on the R/W filemode specified as a result of a previous edit session that ended abnormally, you will receive the message:

   ```
   024E File XEDTEMP CMSUT1 fm already exists
   ```

   You can use the CMS TYPE command to examine the existing file. If you decide you want to keep it, use the CMS RENAME command to give it a new file ID. If the file is not valid or incomplete, you may erase it.
8. To use the automatic save function on a file on a minidisk, you must have the minidisk accessed in read/write mode. If the file is in a directory, you must own the directory and have it accessed in read/write mode.
9. If the file being edited is empty when an autosave occurs, you will receive an error message and the empty file will not be saved.
10. The automatic save function may be used for a byte file system (BFS) file. However, the file is written to an SFS file or a file on a minidisk. (Note, the maximum record length for variable files is 65 535.)

Examples

In the following example, the SET AUTOSAVE subcommand writes the file to a disk or SFS directory every time you change 15 lines.

```
set autosave 15
```

Notes for Macro Writers

1. You can use the EXTRACT/AUTOSAVE/ subcommand to return the autosave information for further processing by a macro.

Responses

Each time an automatic save occurs, the editor writes the file to your file mode A (or the mode specified in the SET AUTOSAVE subcommand). It assigns the file a unique numeric file name and a file type of AUTOSAVE. QUERY UNIQUEID displays the unique AUTOSAVE file name.

The identifier has the form *rrrnnnnn* where *rrr* is the number XEDIT associates with the ring and *nnnnn* is the current autosave number. When the ring number (*rrr*) is less than 100, leading zeros are dropped. Consequently, the file name can have as many as eight digits or as few as six, depending on the ring number.

If no other AUTOSAVE file exists, the file name and file type are:

```
100001  AUTOSAVE
```

In this example, the left-most 1 is the ring number and the right-most 1 is the current autosave number.

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

When an autosave occurs, the following message is displayed:

```
510I Autosaved as fn ft fm
```

Messages and Return Codes

024E File XEDTEMP CMSUT1 *fm* **already exists [RC=28]**
037E Filemode *mode* **is accessed as read/only [RC=12]**
048E Invalid filemode *mode* **[RC=24]**
069E Filemode *mode* **not accessed [RC=36]**
520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]
1350E AUTOSAVE must be targeted to your own directory [RC=12]

where return codes are:

0	Normal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Mode is read only; or cannot AUTOSAVE to another user's directory
24	Invalid file mode
28	File XEDTEMP CMSUT1 *fm* already exists
36	Corresponding mode is not accessed

SET BFSLINE

```
►►──┬─────┬──BFSLine──┬─lrecl───────┬──────────────────────────────────────────────►◄
    └─SET─┘           ├─NL──────────┤
                      ├─CRLF────────┤
                      ├─CRNL────────┤
                      ├─/string/────┤
                      └─/hexstring/─┘
```

Purpose

Use the BFSLINE option to change the value specified (or defaulted) by the XEDIT BFSLINE option. This affects the file when it is written back out to the byte file system (BFS). It does not change the file (by reinterpreting end-of-line characters) as it is displayed in the current XEDIT session. It also does not affect a file being written to SFS or a minidisk.

Operands

lrecl
: indicates the file should be treated as a fixed record format file (RECFM=F) with no interpretation of records based on end-of-line characters. No end-of-line characters are added to the file when it is written back out to the BFS if BFSLINE *lrecl* is still in effect. The *lrecl* is limited to eight characters.

NL
: indicates the new line character (X'15') should be used to delineate lines when writing a BFS file.

CRLF
: indicates that carriage return/line feed (X'0D25') should be used to delineate lines when writing a BFS file.

CRNL
: indicates carriage return and new line characters (X'0D15') should be used to delineate lines when reading or writing a BFS file.

/string/
: allows the user to specify 1 - 2 characters that are used to delineate lines when reading or writing a BFS file. These characters are translated to uppercase before they are utilized. The slash character (/) may not be one of these characters (in other words, you cannot enter ///).

/hexstring/
: specifies a hexadecimal string of 2 or 4 characters that defines the value to be used for BFSLINE. The *hexstring* must be in the format X'nnnn' or X'nn'. You must not specify any spaces in the string, and there must be 2 or 4 hexadecimal characters in the string.

Initial Setting

The initial setting is `BFSLINE NL` (or as specified in the XEDIT command or the LOAD subcommand).

Usage Notes

1. Changing the BFSLINE value affects how the file is stored in the BFS. For example:
   - If you begin the XEDIT session with a value of BFSLINE NL, all new line characters (X'15') are stripped from the file while in the XEDIT session. Changing to BFSLINE 80 and entering FILE would save your file; no end-of-line characters would be reinserted in the file. If RECFM is still variable (V), trailing blanks are stripped from each line of the file. If RECFM has been changed to F, no trailing blanks are removed and each line is padded with blanks up to the length of the LRECL value.
   - If you begin the XEDIT session with a value of BFSLINE 40, no end-of-line characters are stripped from the file. It is presented as it is stored in the BFS, broken into 40 character records. If you change to BFSLINE CRLF, X'0D25' will be inserted after each 40 characters when the file is written to the BFS. If RECFM is V, trailing blanks are stripped from each line.

     When specifying a BFSLINE value for use on files containing DBCS characters, ensure you use a value that will not conflict with DBCS characters. The hexadecimal code for a DBCS character must be X'00', X'40', or in the range of X'41' to X'FE'.

   As you can see, changing the RECFM setting also affects how the file is stored in the BFS.
2. BFSLINE has no effect when a PUT, PUTD, GET, FILE, or SAVE subcommand is used to read or write an SFS or minidisk file when NAMETYPE CMS is in effect.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]
554E Not enough virtual storage available [RC=104]

where return codes are:

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

SET BRKKEY

```
                                          ┌─PA1─┐
►►──┬─────┬──BRKkey──┬──ON──────┼─────┼──┬────────────────────────────────────────────────►◄
    └─SET─┘          │          └─key─┘  │
                     └──OFF──────────────┘
```

Purpose

Use the SET BRKKEY option to inform the editor whether CP should break in when the *BRKKEY* (defined by CP TERMINAL BRKKEY) is pressed.

Operands

ON
indicates pressing the key defined as the BRKKEY causes a control break-in by CP. If you set BRKKEY ON without assigning it to a specific key, XEDIT sets BRKKEY ON PA1 by default.

OFF
indicates no control break-in is desired and XEDIT can use the definition of the key that was depressed. You can then use this key in XEDIT as it has been defined by the SET PF*n* or SET PA*n* subcommand.

key
is the PA1 key or any PF key (1-24). PA1 is the default. Be sure the key you specify is available on the terminal you are using.

Initial Setting

Defined by the CP BRKKEY setting.

Usage Notes

1. CP initially defines the BRKKEY as the PA1 key. For more information on how to define a BRKKEY, see the TERMINAL command in *z/VM: CP Commands and Utilities Reference*.
2. Issuing the editor SET BRKKEY subcommand changes the CP TERMINAL BRKKEY setting. (The new setting remains in effect after you leave the XEDIT session.)
3. If the BRKKEY is set to ON and you enter full-screen CMS, the BRKKEY setting is changed to OFF.
4. If you do not have either a physical or virtual console, the BRKKEY setting will be interpreted as OFF. If you are in a GUI environment, the BRKKEY setting will be interpreted as OFF. Use your terminal emulator to cause a control break-in by CP.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
525E	**Invalid PF key/PA key number [RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal

5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET CASE

```
                                       (1)
                                    ┌──────────┐
►►──┬───────┬──CASE──┬─Uppercase─┬──┤          ├────────────────────────────────────────►◄
    └──SET──┘        └─Mixed─────┘  ├─Respect──┤
                                    └─Ignore───┘
```

Notes:

1 If Respect or Ignore is not specified, the default is the previous setting.

Purpose

Use the CASE option to control how letters are entered into the file, how commands are translated from the command line (whether the commands are entered from the terminal or from the console stack), and whether uppercase and lowercase letters are significant in target searches.

Operands

Uppercase
indicates the editor is to translate all lowercase letters to uppercase (whether the lines are entered from the terminal or from the console stack).

Mixed
indicates the editor is not to translate uppercase and lowercase letters (whether the lines are entered from the terminal or console stack).

Respect
In target searches, an uppercase letter does not match a lowercase letter (and a lowercase letter does not match an uppercase letter). For example:

```
/This Text/
```

does not locate in the file:

```
this text
```

Ignore
In target searches, uppercase and lowercase representations of the same letter match. For example:

```
locate /THIS TEXT/
```

locates in the file:

```
this text
```

Initial Setting

R is always the initial setting; M or U is based on file type. See Appendix A, “File Type Defaults,” on page 453.

Examples

In the following example, the SET CASE subcommand tells the editor to ignore the difference between uppercase and lowercase.

```
set case mixed ignore
```

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET CMDLINE

```
►►──┬─────┬──CMDline──┬─On─────┬──────────────────────────────────────────────────►◄
    └─SET─┘           ├─OFf────┤
                      ├─Top────┤
                      └─Bottom─┘
```

Purpose

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

Operands

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

OFf
removes the command line from the screen.

Top
defines the second line of the screen as the command line.

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

Initial Setting

```
CMDLINE  ON
```

Usage Notes

1. When CMDLINE is set to TOP, BOTTOM, or OFF, no XEDIT status area is displayed.
2. Before setting CMDLINE OFF, you should establish a method of resetting the CMDLINE, perhaps by setting a PF key or through the use of a prefix macro, if you wish to use the command line again.
3. The command line cannot be overlayed by a reserved line unless it is first set OFF.
4. If CMDLINE ON is in effect and you split the screen vertically (see “SET SCREEN” on page 329), only one line is available as a command line when two views are adjacent. However, when you return to a single screen, both lines are again available. SET CMDLINE ON is not valid when issued from a vertical split screen.
5. With CMDLINE TOP and the default SET MSGLINE setting (line 2), a message overlays the command line, including the arrow. Press CLEAR (or any key that does not produce a message) to restore the command line. To avoid this situation, assign the message line to line 1 or line 3 (see “SET MSGLINE” on page 289) when using CMDLINE TOP.
6. When SET CMDLINE is entered with the ON, TOP, or BOTTOM operands, the cursor is positioned in the first column of the command line.

Examples

For more information, see *z/VM: XEDIT User’s Guide*.

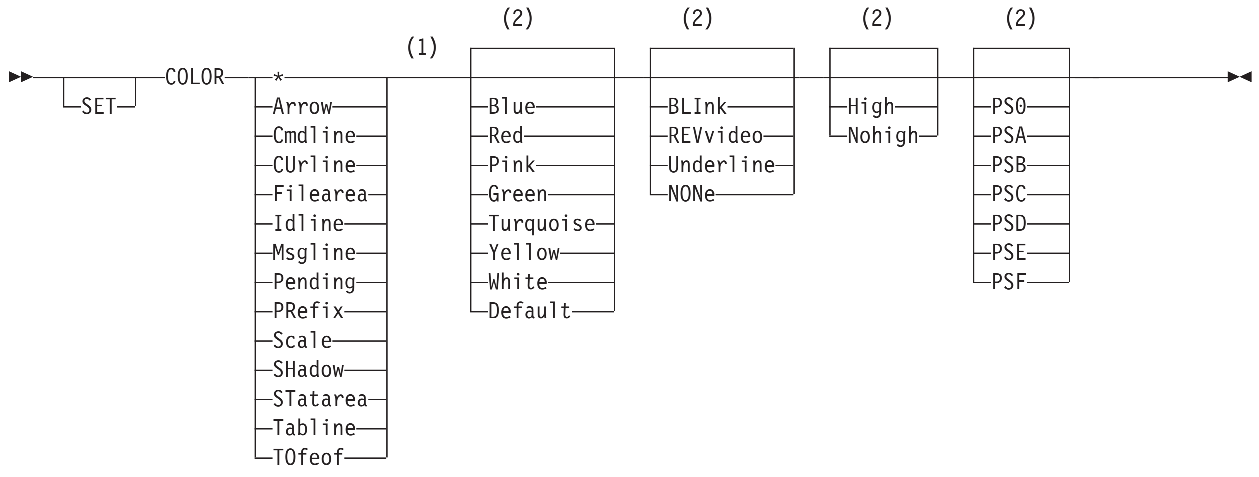
Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
526E	**Option** *option* **valid in display mode only [RC=3]**
545E	**Missing operand(s) [RC=5]**
568E	**Subcommand not valid with this screen definition [RC=5]**

where return codes are:

0	Normal
3	Operand is valid only for display terminal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET COLOR



Notes:

1 One of the operands must be chosen.

2 The default is the previous setting.

Purpose

Use the COLOR option to associate specific colors with certain areas of the XEDIT screen. Note, this option may be specified as COLOR or COLOUR.

Operands

* indicates all of the fields will be chosen in that group. (Arrow, Cmdline, CUrline, Filearea, Idline, Msgline, Pending, PRefix, Scale, SHadow, STatarea, Tabline, and TOfeof).

Arrow
the arrow pointing to command line.

Cmdline
the line where commands are entered.

CUrline
the file line that is the current line.

Filearea
file data area, excluding the current line, the TOF and EOF lines.

Idline
the file identification line on line 1 of the screen.

Msgline
the area for message display.

Pending
the pending macro or subcommand as entered in the prefix area.

PRefix
the five-position area that lies either to the left or the right of each line of the file, as defined in SET PREFIX.

Scale
the scale line.

SHadow
shadow line(s) resulting from selective line editing (see Appendix B, "Effects of Selective Line Editing Subcommands," on page 455).

STatarea
status area in lower right-hand corner of screen, which displays the current status of your editing session.

Tabline
the line that displays a T in every tab column, according to the current tab settings.

TOfeof
the top of file and end of file notices.

Blue
Red
Pink
Green
Turquoise
Yellow
White
Default
are the choices for the color of the field. If a color is not specified, the current color is not changed.

BLInk
REVvideo (reverse video)
Underline
NONe
are the choices for the extended highlighting of the field. If an extended highlighting setting is not specified, the current extended highlighting setting is not changed.

High
displays the field highlighted.

Nohigh
displays the field not highlighted (normal intensity).

If you omit HIGH or NOHIGH, the current HIGH or NOHIGH setting is not changed. See Usage Note 1 on page 255.

PS0
PSA
PSB
PSC
PSD
PSE
PSF
are the choices for the programmed symbol set of the field. PS0 indicates using the default character set.

Initial Setting

```
FIELD      COLOR     EXTHI  HIGH/NOHIGH  PSs

ARROW      DEFAULT   NONE   HIGH         PS0
CMDLINE    DEFAULT   NONE   NOHIGH       PS0
```

```
CURLINE    DEFAULT   NONE   HIGH         PS0
FILEAREA   DEFAULT   NONE   NOHIGH       PS0
IDLINE     DEFAULT   NONE   HIGH         PS0
MSGLINE    RED       NONE   HIGH         PS0
PENDING    DEFAULT   NONE   HIGH         PS0
PREFIX     DEFAULT   NONE   NOHIGH       PS0
SCALE      DEFAULT   NONE   HIGH         PS0
SHADOW     DEFAULT   NONE   NOHIGH       PS0
STATAREA   DEFAULT   NONE   HIGH         PS0
TABLINE    DEFAULT   NONE   HIGH         PS0
TOFEOF     DEFAULT   NONE   NOHIGH       PS0
```

Usage Notes

1. The SET COLOR subcommand accepts the color, extended highlighting and programmed symbol set operands regardless of whether the device has the ability to use those attributes. However, the action taken is dependent upon the device. For example, HIGH and NOHIGH are ignored on a 3279 display (unless the color was DEFAULT), color is ignored on a 3278 display, and color, extended highlighting, and character set are ignored on a 3277 display. Also, QUERY and EXTRACT return all the current settings, even those ignored on any given terminal.
2. The attributes following the field keyword can appear in any order. Only one attribute is required following the field.
3. If TOFEOF is set OFF (see “SET TOFEOF” on page 361), the extended highlighting attributes (reverse video and underline) are still visible.
4. The color and extended highlighting of the current line override the color and extended highlighting of the filearea and TOFEOF for any line that is the current line.
5. The color and extended highlighting of the scale override the color and extended highlighting of the tabline when the scale and tabline are on the same line.
6. On 3270 terminals equipped with the Programmed Symbol (PS) feature you can specify alternate character sets to be used on your terminal. The characters in these sets use different symbols, such as a different style or font, than the default character set.

   New character sets are loaded using the Graphic Data Display Manager (GDDM) licensed program (5748-XXH) or an application that loads programmed symbol sets. During the loading of programmed symbol sets, the user specifies symbol set identifiers (SSIDs). In XEDIT, the allowable SSIDs are in the decimal range 193 through 198. These SSIDs correspond to PSA through PSF, respectively (that is, 193 to PSA). Load character sets before invoking XEDIT. Otherwise, if you load a new set, it is not recognized and the last character set loaded prior to invoking the editor is displayed; likewise, if you drop a programmed symbol set, the editor is not aware the set is no longer available and attempts to use that set. This may cause I/O errors when the display is written (resulting in SET TERMINAL TYPEWRITER). To avoid problems, do not load or delete program symbol sets when in the XEDIT environment.

   For more information on programmed symbol sets, see *IBM 3270 Information Display System Data Stream Programmer’s Reference*.
7. If you are in a GUI environment, BLINK, PSS, and UNDERLINE are ignored. Also, workstation color changes do not affect the values in SET COLOR.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET COLPTR

```
►►──┬─────┬──COLPtr──┬─ON──┬──────────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

Use the COLPTR option only with typewriter terminals to control whether the column pointer (represented as an underscore character) is displayed.

Operands

ON
displays the column pointer.

OFF
removes the column pointer from the display.

Initial Setting

`COLPTR ON` (typewriter terminals only)

Usage Notes

1. To display the column pointer, your typewriter terminal must be equipped with a backspace key.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET CTLCHAR

```
►►──┬─────┬──CTLchar──┬──────char──┬─┤ Control Character Options ├─┬─┬──────────►◄
    └─SET─┘           │            ├─OFF───────────────────────────┤ │
                      │            └─Escape────────────────────────┘ │
                      └─OFF──────────────────────────────────────────┘
```

Control Character Options:

```
                  ┌─Default───┐  ┌─NONe─────┐  ┌─Nohigh────┐  ┌─PS0─┐
├──┬─Protect───┬──┼───────────┼──┼──────────┼──┼───────────┼──┼─────┼──────────────────┤
   └─Noprotect─┘  ├─Blue──────┤  ├─BLInk────┤  ├─High──────┤  ├─PSA─┤
                  ├─Red───────┤  ├─REVvideo─┤  └─Invisible─┘  ├─PSB─┤
                  ├─Pink──────┤  └─Underline┘                 ├─PSC─┤
                  ├─Green─────┤                               ├─PSD─┤
                  ├─Turquoise─┤                               ├─PSE─┤
                  ├─Yellow────┤                               └─PSF─┘
                  └─White─────┘
```

Purpose

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

- Color
- Extended highlighting
- Highlighting
- Programmed symbol set
- Protection
- Visibility

This option is designed to be issued from a macro, particularly when you are creating display panels and other interactive information. It applies to lines reserved by the SET RESERVED subcommand.

Operands

char
is any character, uppercase or lowercase, interpreted as a control character when embedded in the text to be displayed. If specified with the ESCAPE operand, this character identifies the next character in the text as a control character. No more than 63 characters can be defined.

OFF
resets all control characters that have been defined (SET CTLCHAR OFF), or resets a specified character (SET CTLCHAR *char* OFF).

Protect
specifies the string following *char* is to be displayed in a protected field.

Noprotect
specifies the string following *char* is to be displayed in an unprotected field.

Escape
: specifies 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.

Blue
Red
Pink
Green
Turquoise
Yellow
White
Default
: are the choices for the color of the field. If a color is not specified, the default is DEFAULT (the default base color).

BLInk
REVvideo (reverse video)
Underline
NONe
: are the choices for the extended highlighting of the field. If an extended highlighting setting is not specified, the default is NONE (no extended highlighting).

High
: specifies the string following *char* is to be displayed highlighted. NOHIGH is the default.

Nohigh
: specifies the string following *char* is to be displayed not highlighted (normal intensity). If you omit HIGH, NOHIGH, or INVISIBLE, NOHIGH is assigned.

Invisible
: defines a field where the characters are not displayed, for example, a password. If you omit HIGH or INVISIBLE, NOHIGH is the default.

PS0
PSA
PSB
PSC
PSD
PSE
PSF
: are the choices for the programmed symbol set of the field. PS0 indicates using the default character set.

Initial Setting

No control characters are defined. CTLCHAR is set OFF.

Usage Notes

1. If you are in a GUI environment, BLINK, PSS, and UNDERLINE are ignored.

Notes for Macro Writers

1. Using control characters to specify the display within a reserved line requires three steps:
   a. Define the control character identifier using the ESCAPE operand of SET CTLCHAR.

   b. Define the control character with its associated features.
   c. Use the SET RESERVED subcommand with the control character identifier and control character to specify the display within the reserved line.

   See “Examples” for an example of each of these steps.

2. If you use SET CTLCHAR to set up multiple fields within reserved lines, particularly input (unprotected) fields, you should use READ with the TAG option. Multiple fields on a single reserved line are stacked separately, and a tag identifying the origin of each line is inserted. For more information on using READ with the TAG option, see “READ” on page 204.
3. The SET CTLCHAR subcommand accepts the color, extended highlighting, and Programmed Symbol Set operands whether the terminal has the ability to use those attributes. However, the action taken depends upon the device. For example, HIGH and NOHIGH are ignored on a 3279 display (unless the color was DEFAULT), color is ignored on a 3278 display, and color, extended highlighting, and character set are ignored on a 3277 display. Also, QUERY and EXTRACT return all the settings, even those ignored on any given terminal.
4. On 3270 terminals equipped with the Programmed Symbol (PS) feature you can specify alternate character sets to be used on your terminal. The characters in these sets use different symbols, such as a different style or font, than the default character set.

   New character sets are loaded using the Graphic Data Display Manager (GDDM®) licensed program (5748-XXH) or an application that loads programmed symbol sets. Load character sets before invoking XEDIT. Otherwise, if you load a new set, it is not recognized and the last character set loaded before invoking the editor is displayed; likewise, if you drop a programmed symbol set, the editor is not aware the set is no longer available and attempts to use that set. This may cause I/O errors when the display is written (resulting in SET TERMINAL TYPEWRITER). To avoid problems, do not load or delete program symbol sets when in the XEDIT environment.

   For more information on programmed symbol sets, see *IBM 3270 Information Display System Data Stream Programmer’s Reference*.

Examples

This section shows examples of the SET CTLCHAR subcommand.

Example 1

In this example, ! is defined as a control character identifier. When ! appears in the text, the next character is interpreted as a control character.

```
set ctlchar ! escape
```

Example 2

In this example, when % appears in the text, preceded by the control character identifier (!), the data that follows it is displayed protected and highlighted.

```
set ctlchar % protect high
```

Example 3

In this example, when " appears in the text, preceded by the control character identifier (!), the data that follows it is displayed protected and not highlighted.

```
set ctlchar " protect nohigh
```

Example 4

In this example, when ? appears in the text, preceded by the control character identifier (!), the data that follows it is protected, underlined, and displayed in blue using programmed symbol set A.

```
set ctlchar ? protect blue underline psa
```

Example 5

In this example, when the screen is displayed, the word *example* is highlighted on line 3 and the word *highlighting* is protected, underlined, and displayed in blue using programmed symbol set A (PSA).

```
set reserved 3 noh This is an !%example!" of selective !?highlighting!"
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]
695E Cannot define more than 63 CTLCHARs [RC=4]

where return codes are:

0	Normal
4	Too many control characters defined
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET CURLINE

```
►►──┬─────┬──CURLine──ON──┬─M───┬──────────────────────────────────────►◄
    └─SET─┘               ├─Mn──┤
                          ├─ M+n ─┤
                          ├─ M−n ─┤
                          ├─n───┤
                          ├─ +n ─┤
                          └─ −n ─┘
```

Purpose

Use the CURLINE option to define the position of the current line on the screen.

Operands

ON
displays the current line on the screen.

M specifies the line in which the current line is displayed. The M stands for *middle of the screen* (rounded up for odd sized screens).

M*n*
M+*n*
M–*n*
You can combine M with a constant positive (+ is implicit) or negative integer to mean *n* lines below the middle of the screen (M+*n*), or *n* lines above the middle of the screen (M–*n*). For example, CURLINE ON M means the *middle of the screen*, and CURLINE ON M+3 means *three lines below the middle of the screen*.

n
+*n*
–*n* specifies the line in which the current line is displayed. The *n* or +*n* (+ is implicit) specifies the current line is displayed *n* lines from the top of screen; –*n* specifies the current line is displayed *n* lines from the bottom of screen. For example, CURLINE ON –3 means the current line is three lines from the bottom of the screen.

Initial Setting

`CURLINE ON M` (middle of the screen)

Usage Notes

1. Under certain conditions, the current line as it appears on your display screen does not have the same line number you specified on the SET CURLINE subcommand. This occurs when the CURLINE line setting is the same as the line setting for any of these SET options:
   - SET RESERVED
   - SET SCALE
   - SET TABLINE
   - SET MSGLINE

2. When the CURLINE line setting is the same as the line setting for SET SCALE, for example, the scale occupies the specified line and the current line is moved down to the next available line.

   Note: Other subcommands that refer to CURLINE (EXTRACT, for example) use the true current line regardless of where it happens to appear on the display screen.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
521E	**Invalid line number [RC=5]**
526E	**Option** *option* **valid in display mode only [RC=3]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
3	Operand is valid only for display terminal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET DISPLAY

```
                       ┌─n1─┐
►►──┬─────┬─DISPlay─n1─┼────┼──────────────────────────────────────────►◄
    └─SET─┘            ├─n2─┤
                       └─*──┘
```

Purpose

Use the DISPLAY option to specify which selection levels of lines (as defined in SET SELECT) are displayed. For an example of how to combine SET DISPLAY effectively with the other selective line editing subcommands (SET SELECT, SET SCOPE, and SET SHADOW), see "SET SELECT" on page 333.

Operands

n1 is a positive whole number that represents a selection level of lines you wish to display. If you specify *n1* alone, only the lines having a selection level of *n1* are included in the display.

n2 is a positive whole number that represents a selection level of lines. Specifying both *n1* and *n2* includes in the display all lines with selection levels between *n1* and *n2* inclusive. The value of *n2* must be equal to or greater than that of *n1*.

***** Specifying an asterisk (*) instead of *n2* displays all lines with a selection level of *n1* or greater.

Initial Setting

```
DISPLAY 0 0
```

Usage Notes

1. To display the entire file, enter SET DISPLAY 0 *. Then, before you can use the X prefix macro to exclude lines, you must reset the DISPLAY level to something other than asterisk (*).
2. For examples of this and other selective line editing subcommands, you can examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display. See also Appendix B, "Effects of Selective Line Editing Subcommands," on page 455.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
545E Missing operand(s) [RC=5]
600E First selection level (*nn***) cannot be greater than the second selection level (***nn***) [RC=5]**

where return codes are:

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

SET ENTER

```
                     (1)        (2)
                  ┌────────┐ ┌─────────┐
►►──┬─────┬──ENTer┴────────┴─┴─────────┴────────────────────────────────────────────►◄
    └─SET─┘       ├─BEFORE─┤ ├─string──┤
                  ├─AFTER──┤ ├─NULLKEY─┤
                  ├─ONLY───┤ ├─COPYKEY─┤
                  └─IGNORE─┘ └─TABKEY──┘
```

Notes:

1 Refer to the BEFORE operand description for the default.

2 If *string* is not specified, ENTER is undefined.

Purpose

Use the ENTER option to define a meaning for the keyboard ENTER or to remove the meaning associated with ENTER.

Operands

BEFORE
specifies the key definition is executed before the contents of the command line. BEFORE is the default for SET ENTER, unless the *string* begins with ? or =. For these two strings, ONLY is the default.

AFTER
specifies the key definition is executed after the contents of the command line.

ONLY
specifies only the key definition is executed, thereby ignoring the command line.

IGNORE
specifies the key definition is ignored when you enter something on the command line, thereby only executing the command line.

string
is any XEDIT subcommand or macro (including CP, CMS, or EXEC), which is executed when you press ENTER. If *string* is null, ENTER is undefined.

NULLKEY
When you press ENTER, blank characters are changed to nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, blanks are changed to nulls on the file line with which that prefix area is associated.

COPYKEY
When you press ENTER, the exact content of the virtual screen is copied into the printer spool.

TABKEY
When you press ENTER, the cursor is moved into the next tab column, as defined in the SET TABS subcommand.

Initial Setting

```
ENTER IGNORE COMMAND CURSOR CMDLINE 1 PRIORITY 30
```

Usage Notes

1. When you use ENTER set to COPYKEY, you should close the printer at the end of the editing session. If an error occurs in printing the screen, no copy is placed in the printer and only severe error messages are issued.
2. When you press ENTER set to TABKEY, COPYKEY, or NULLKEY, no screen changes are processed, including the command line, and the keywords BEFORE, AFTER, ONLY, and IGNORE have no effect. A PA/PF key would have to be used to process the command line or to change the ENTER key definition.
3. ONLY as part of the ENTER key setting should be used with extreme care, because the command line is not processed. A PA/PF key would have to be used to process the command line or to change the ENTER key definition.
4. When the ENTER key is pressed, the ENTER key definition, the contents of the command line, or both are placed on the console stack in the order described in "Using READ CMDLINE" on page 205. Thus if BEFORE is specified, execution of the ENTER key definition may alter or suppress execution of the command line by altering the contents of the stack; similarly, if AFTER is specified, execution of the command line may alter or suppress execution of the ENTER key definition.
5. If you are in a GUI environment, NULLKEY is ignored.

Examples

In the following example, setting the priority to 5 prevents the cursor from moving to the command line if you are in the file area when you press ENTER.

```
set enter ignore command cursor cmdline 1 priority 5
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET ESCAPE

```
                              (1)
                            ┌───────┐
►►──┬─────┬─ESCape──┬─ON──┬─┼───────┼───────────────────────────────────────────►◄
    └─SET─┘         └─OFF─┘ └─char──┘
```

Notes:

1 If *char* is not specified, the default is the previous setting.

Purpose

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.

Operands

ON
turns on the use of an escape character.

OFF
turns off the use of an escape character.

char
specifies a character to be used.

Initial Setting

`ESCAPE  ON` (disconnected or typewriter terminal)
`ESCAPE OFF` (display terminal)

A default escape character is assigned according to file type (the '/' character for most file types); see Appendix A, “File Type Defaults,” on page 453.

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 with QUERY ESCAPE.
3. This subcommand has no meaning on a display terminal.
4. You can specify the escape character in hexadecimal if SET HEX ON is in effect.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET ETARBCH

```
                                   (1)
►►──┬─────┬──ETARBCH──┬─ON──┬──┬──────┬──────────────────────────────────►◄
    └─SET─┘           └─OFF─┘  └─char─┘
```

Notes:

1 If *char* is not specified, the default is the previous setting.

Purpose

Use the ETARBCH option to define an extended arbitrary character within a Double-Byte Character Set (DBCS) string. An extended arbitrary character used between two character strings in a target definition means *ignore all intervening characters when searching for a match in the file.*

Operands

ON
turns on the use of a double-byte special character as an extended arbitrary character.

OFF
turns off the use of a special character as an extended arbitrary character without changing the current character definition.

char
specifies the double-byte character to be used as the extended arbitrary character.

Initial Setting

The initial setting is a double-byte monetary symbol (X'425B').

`ETARBCH OFF`	`"]$"`	(X'425B') on a 3278 and 3279 terminal
`ETARBCH OFF`	`""$"`	(X'425B') on a 3277 terminal
`ETARBCH OFF`	￥	(X'425B') on a 5550 Multistation

Usage Notes

1. See Appendix F, “Using Double-Byte Character Sets” on page 467 for information on using DBCS strings in the XEDIT environment.
2. Extended arbitrary characters can be used in targets and in the CHANGE, COUNT, and SPLIT subcommands.
3. You can specify the extended arbitrary character in hexadecimal if SET HEX ON is in effect.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET ETMODE

```
►►──┬─────┬──ETMODE──┬─ON──┬───────────────────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

Use the ETMODE (extended mode) option to specify XEDIT should recognize Double-Byte Character Set (DBCS) strings. In this mode, XEDIT can manipulate and display DBCS strings on terminals that support Double-Byte Character Sets.

Operands

ON
specifies XEDIT recognizes and manipulates DBCS strings that begin with a shift-out character and end with a shift-in character.

OFF
specifies all data is treated as 1-byte EBCDIC data.

Initial Setting

The initial setting is based on whether the terminal can display double-byte characters. If it can, the initial setting is ON; if not, the setting is OFF.

Usage Notes

1. If full-screen CMS is off or suspended and you disconnect from a terminal that supports DBCS and reconnect to a terminal that does not support DBCS, the first time you enter the XEDIT command on the new terminal, ETMODE is set ON. If you disconnect from a terminal that does not support DBCS and reconnect to a terminal that does, the first time you enter the XEDIT command on the new terminal, ETMODE is set OFF.
2. See Appendix F, "Using Double-Byte Character Sets," on page 467 for information on using DBCS strings in the XEDIT environment.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET FILLER

```
                        (1)
                   ┌───────┐
►►──┬─────┬─FILler─┤       ├──────────────────────────────────────────────►◄
    └─SET─┘        └─char──┘
```

Notes:

1 If *char* is not specified, the default filler character is a blank.

Purpose

Use the FILLER option to define a character for the editor to use when a line is expanded; that is, when tab characters are removed and replaced by an appropriate number of filler characters (see "EXPAND" on page 86).

Operands

char
specifies a filler character to be used. The default filler character is a blank.

Initial Setting

The *blank* character is defined as the filler.

Usage Notes

1. You can specify the filler character in hexadecimal if SET HEX ON is in effect.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET FMODE

```
►►──┬─────┬──FMode──fm────────────────────────────────────────────────────────►◄
    └─SET─┘
```

Purpose

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

Operands

fm is the new file mode. You can specify a file mode letter (A through Z) or a file mode letter and number (0-6). If you specify only a file mode letter, the existing file mode number is used.

Initial Setting

Current file mode of file being edited.

Usage Notes

1. The specified file mode is used the next time you enter FILE or SAVE. If the file being edited has been written to disk or directory before, that copy of the file remains unchanged.
2. If the disk or directory the file mode specifies already contains a file with the same file name and file type and you enter FILE or SAVE, the editor displays an error message.
3. If the file mode specified is that of a read-only minidisk and you enter FILE or SAVE, the editor displays an error message.
4. If you are editing a member of a MACLIB, this command changes the mode of the member.
5. If you are editing a byte file system (BFS) file, the initial setting for FMODE is all blanks. SET FMODE is allowed if you are editing a BFS file. However, it is not used until SET NAMETYPE CMS is in effect.

Messages and Return Codes

048E	**Invalid filemode** *mode* **[RC=24]**
069E	**Filemode** *mode* **not accessed [RC=36]**
520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
555E	**File** *fn ft fm* **already in storage [RC=4]**

where return codes are:

0	Normal
4	File already in storage
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
24	Invalid file mode
36	Corresponding mode not accessed

SET FNAME

```
►►──┬─────┬──FName──fn──────────────────────────────────────────────────►◄
    └─SET─┘
```

Purpose

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

Operands

fn is the new file name. It can be from 1 - 8 characters and will be translated to uppercase.

Initial Setting

Current file name of file being edited.

Usage Notes

1. The specified file name is used the next time a FILE or SAVE is issued. If the file being edited has been written to disk or directory before, that copy of the file remains unchanged.
2. If a file already exists with the specified file name and the same file type and file mode and you issue FILE or SAVE, the editor displays an error message.
3. If you are editing a member of a MACLIB, this command changes the member name.
4. If you are editing a byte file system (BFS) file, the initial setting for FNAME is all blanks. SET FNAME is allowed if you are editing a BFS file. However, it is not used until SET NAMETYPE CMS is in effect.

Messages and Return Codes

062E	**Invalid character in fileid** *fn ft fm* **[RC=20]**
520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
555E	**File** *fn ft fm* **already in storage [RC=4]**

where return codes are:

0	Normal
4	File already in storage
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
20	Invalid character in the string proposed as file name

SET FTYPE

```
►►──┬─────┬──FType──ft──────────────────────────────────────────────────►◄
    └─SET─┘
```

Purpose

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

Operands

ft	is the new file type. It can be from 1 - 8 characters and will be translated to uppercase.

Initial Setting

Current file type of file being edited.

Usage Notes

1. The specified file type is used the next time a FILE or SAVE is issued. If the file being edited has been written to disk or directory before, that copy of the file remains unchanged.
2. If a file already exists with the specified file type and the same file name and file mode and you issue FILE or SAVE, the editor displays an error message.
3. When editing a member of a MACLIB, the file type must remain MEMBER.
4. If you are editing a byte file system (BFS) file, the initial setting for FTYPE is all blanks. SET FTYPE is allowed if you are editing a BFS file. However, it is not used until SET NAMETYPE CMS is in effect.

Messages and Return Codes

062E	**Invalid character in fileid** *fn ft fm* **[RC=20]**
520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
555E	**File** *fn ft fm* **already in storage [RC=4]**

where return codes are:

0	Normal
4	File already in storage
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
20	Invalid character in the string proposed as file type

SET FULLREAD

```
►►──┬─────┬──FULLread──┬─ON──┬──────────────────────────────────────────────►◄
    └─SET─┘            └─OFF─┘
```

Purpose

Use the FULLREAD option to allow XEDIT and CMS to recognize 3270 null characters in the middle of screen lines. SET FULLREAD changes the CMS setting of the FULLREAD option.

Operands

ON
in combination with SET NULLS ON, lets XEDIT and full-screen CMS recognize nulls in the middle of lines, allowing you to enter tabular or pictorial data without worrying about its being *crushed to the left.* NULLS ON allows you to use insert mode.

OFF
inhibits transmission of nulls from the terminal to XEDIT. The SET NULLS subcommand can in this situation control the relationship between nulls and blanks in screen data.

Initial Setting

Defined by the CMS setting.

Usage Notes

1. When FULLREAD ON is issued, nulls at the end of screen lines part of a logical line occupying more than one physical screen line are dropped. This lets you delete characters in a screen line and still have the reconstructed line flush together even though multi-line 327X lines do not *wrap* when the character delete key (or the insert mode key) is used.
2. Using FULLREAD ON has performance implications. For local channel attached display controllers with several users taking advantage of the SET FULLREAD usability improvement, there is a potential for a significant increase in system response time. For remote attached display controllers, setting FULLREAD ON results in a noticeable increase in response time. Also, because of increased line traffic, the maximum number of terminals that can be supported on a link may be significantly reduced.

   An alternative to using fullread is the 3274 Entry Assist option. For more information, see *3270 Information Display System 3274 Control Unit Entry Assist RPQ 8K1147 User's Guide*.
3. Setting FULLREAD ON prevents you from losing any screen changes when you press a PA key and a message is displayed on a cleared screen.
4. A certain terminal configuration, which imposes several restrictions on your XEDIT session, occurs when going through a VM/Passthru Facility (5749-RC1) (PVM) 327X Emulator link to another VM system. These PVM links can be identified by an S to the immediate left of the node ID in the PVM selection screen. The restrictions are:
   a. The subcommand SET FULLREAD ON cannot be used.

   b. All PA keys (except for the CP defined TERMINAL BRKKEY) are made non-functional.
5. Changing the editor FULLREAD setting also changes the CMS FULLREAD setting.
6. Use of the (—>) key or any PF key set with the SET PF*n* TABKEY subcommand creates nulls in the middle of the command input area, regardless of the SET NULLS setting. When FULLREAD is OFF, these nulls are *crushed* to the left. When FULLREAD is ON, the nulls are converted to blanks. Because these blanks are treated as user-entered characters, they could affect the column placement of the actual user-entered characters in the file area.
7. If you are in a GUI environment, SET FULLREAD is ignored.

Examples

In the following example, (with NULLS ON and FULLREAD OFF) if you enter the following line, using the (—>) key to move the cursor between the columns of data:

```
Vermont     Maine     California     Georgia
```

the data is *crushed* to the left as follows when you press ENTER.

```
VermontMaineCaliforniaGeorgia
```

Nulls are not recognized in the middle of screen lines with NULLS ON and FULLREAD OFF.

However, with NULLS ON and FULLREAD ON, if you enter the same line, using the (—>) key between the columns of data, the nulls are handled as blanks:

```
Vermont     Maine     California     Georgia
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET HEX

```
►►──┬─────┬──HEX──┬─ON──┬──────────────────────────────────────────────────────────────►◄
    └─SET─┘       └─OFF─┘
```

Purpose

Use the HEX option to allow subcommand string operands and targets to be specified in hexadecimal notation. The editor searches for their EBCDIC equivalent.

Operands

ON
allows subcommand string operands and targets to be specified in hexadecimal notation.

For example:

```
locate /X'C1C2C3'/
```

is the same as

```
locate /ABC/
```

OFF
turns off the ability to specify string operands and targets in hexadecimal notation.

Initial Setting

```
HEX  OFF
```

Usage Notes

1. When using ARBCHAR, the arbitrary character must be specified in hex. In the following example, the hex value of $ is 5B.

   ```
   set arbchar on $
   set hex on
   change /x'C15BC2'/x'C3'/
   ```

   is the same as

   ```
   change /A$B/C/
   ```

Messages and Return Codes

520E Invalid operand: ***operand*** **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET IMAGE

```
►►──┬─────┬──IMage──┬─ON────┬──────────────────────────────────────────────────────►◄
    └─SET─┘         ├─OFF───┤
                    └─Canon─┘
```

Purpose

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).

Operands

ON
specifies 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 as a character delete and each column being occupied by the last character typed in. For example:

```
set  tabs  2  5  10
```

INPUT LINE: `XB_TYBZ`

The `B` represents a backspace character.

The `T` represents a tab character.

The _ represents an underscore character.

AFTER PROCESSING: `b_bbZ`

The `b` represents a blank.

OFF
specifies tab and backspace characters are to be left as they are entered.

Canon
specifies tabs are left as they are entered, that is, tabs are not expanded to blank (or filler) characters.

Backspace characters can 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_`

```
 _
 _
```

The `B` represents a backspace character.

The _ represents an underscore character.

AFTER PROCESSING: `_BX_BY_BZ`

This process simplifies searching through the file for a string, because you need not know the exact order which the characters were entered.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults," on page 453.

Usage Notes

1. The IMAGE setting affects the following subcommands and macros:
   - ADD (and A and I prefix subcommands)
   - COMPRESS
   - EXPAND
   - FIND, FINDUP, NFIND, NFINDUP
   - INPUT
   - OVERLAY
   - REPLACE
   - SPLIT
   - VMFOPT
   - All targets
   - Typing over a line

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET IMPCMSCP

```
►►──┬─────┬──IMPcmscp──┬─ON──┬──────────────────────────────────────────►◄
    └─SET─┘            └─OFF─┘
```

Purpose

Use the IMPCMSCP option to control whether subcommands XEDIT does not recognize are transmitted implicitly to CMS and later to CP (if necessary and if CMS SET IMPCP is ON) for execution.

Operands

ON
specifies subcommands the editor does not recognize are sent to CMS and later (if needed and if CMS SET IMPCP is ON) to CP for execution; that is, subcommands unknown to XEDIT are considered to be CMS (or CP) commands.

OFF
specifies unrecognized subcommands are not sent to CMS and CP.

Initial Setting

```
IMPCMSCP  ON
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET LASTLORC

```
                          (1)
                       ┌───────┐
►►──┬─────┬──LASTLorc──┴───────┴───────────────────────────────────────────►◄
    └─SET─┘              └─line─┘
```

Notes:

1 If *line* is not specified, the LASTLORC buffer is set to nulls.

Purpose

Use the LASTLORC (Last Locate or Change) option within a macro to specify the contents of the LASTLORC buffer.

Operands

line
: specifies what you want saved in the LASTLORC buffer. If you omit *line*, the LASTLORC buffer is set to nulls.

Initial Setting

The initial setting of LASTLORC is a null string.

Notes for Macro Writers

1. LASTLORC does not have to be set explicitly. It is normally automatically updated with the last user subcommand when a LOCATE, CLOCATE, CHANGE, or any subcommand in the FIND family is executed.
2. LASTLORC provides a memory capability a macro can use explicitly. For example, SCHANGE uses LASTLORC so a user can repeat the last CHANGE or CLOCATE without having to reenter the data. Likewise a macro that issues a FIND or LOCATE might use this so the user need not retype a command just to continue a search.

Messages and Return Codes

This option issues no messages, it issues return codes as follows:

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

SET LINEND

```
                                  (1)
                                 ┌─────┐
►►──┬─────┬──LINENd──┬─ON──┬────┤      ├──────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘     └─char─┘
```

Notes:

1 If *char* is not specified, the default is the previous setting.

Purpose

Use the LINEND option to determine whether the editor is to recognize a pound sign (#) or other character as a line-end character.

Operands

ON
allows you to enter several adjacent subcommands, separated by the initial character (a pound sign) or a previously specified *char*.

OFF
specifies the editor is not to recognize a line-end character.

char
is the special character to be a line-end character. Use special characters such as @, $, or %. Other characters such as /, a delimiter, or alphabetic characters may cause problems. The initial line-end character is a pound sign (#).

Initial Setting

```
LINEND  ON  #
```

Usage Notes

1. The line-end character may be specified in hexadecimal if SET HEX ON is in effect.
2. For consecutive line-end characters (for example, FILE####) and for a single line-end character that follows the last subcommand entered (for example, FILE#), XEDIT stacks a null line for each line-end character minus 1.
3. In line mode the line-end character from CP is also recognized during your XEDIT session. To set the CP line-end character off or redefine it, issue:
   ```
   cp terminal linend off
           or
   cp terminal linend char
   ```
   with *char* as your choice of CP line-end character. (For more information on the CP line-end symbol, see the TERMINAL command in *z/VM: CP Commands and Utilities Reference*.)
4. If SET CASE UPPERCASE is in effect, the command line is uppercased before it is scanned for line-end characters. This can cause commands with case-sensitive targets to fail to work properly when issued in conjunction with a

SET CASE M R command. For example, issuing SET CASE M R#FIND *xxx* while SET CASE UPPERCASE is in effect causes the *xxx* to be uppercased before the editor searches for it.

Examples

In the following example, entering

```
next 1#change /A/B/#TOP
```

executes as:

```
next 1
change /A/B/
top
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET LRECL

```
►►──┬─────┬──LRecl──┬─n─┬──────────────────────────────────────────────────►◄
    └─SET─┘         └─*─┘
```

Purpose

Use the LRECL option to define a new logical record length, which is used when the file is written to disk, a Shared File System (SFS) directory, or to the byte file system (BFS).

Operands

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.
*****	the logical record length is set to the WIDTH as defined in the XEDIT (or LOAD) command.

Initial Setting

- For SFS and minidisk files, the initial settings are based on the file type. See Appendix A, "File Type Defaults," on page 453.
- For BFS files, the initial setting is whichever is smaller, the WIDTH or the longest record read (if the file exists and is not empty).

Usage Notes

1. The value you specify for *n* must be less than or equal to the value specified for the WIDTH operand of the XEDIT or LOAD subcommand (in the profile). (WIDTH is the amount of virtual storage used for any file line, regardless of its format on disk or directory.)
2. If the new logical record length is smaller than the previous one, data may be lost when the file is written to disk or directory. A smaller logical record length may also create new values for zone settings, the column pointer, and the truncation column. You should verify the following relationships:

   ```
   ZONE1 ≤ COLUMN POINTER ≤ ZONE2 ≤ TRUNC ≤ LRECL ≤ WIDTH
   ```

   The column pointer can also be positioned at TOL (Zone1 – 1) and at EOL (Zone2 + 1).
3. See "LOAD" on page 141 for more details on the initial setting of LRECL during profile execution.

Messages and Return Codes

516E	**LRECL too large for V-format file [RC=4]**
519E	**LRECL must not exceed WIDTH (*nn*) [RC=5]**
520E	**Invalid operand:** *operand* **[RC=5]**
543E	**Invalid number:** *number* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
698W	**New record length may result in loss of double-byte characters [RC=3]**

where return codes are:

0	Normal
3	Records truncated
4	LRECL must be lower than 65 536 for RECFM V when editing an SFS or minidisk file.
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET MACRO

```
►►──SET──MACRO──┬─ON──┬──────────────────────────────────────────────────────────►◄
                └─OFF─┘
```

Purpose

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

Operands

ON
specifies the editor is to look for macros before it looks for subcommands.

OFF
specifies the editor is to look for subcommands before it looks for macros.

Initial Setting

```
MACRO  OFF
```

Usage Notes

1. This SET option can resolve an ambiguous situation, when a macro and a subcommand have the same name. If a synonym is defined for the subcommand, the macro name executes only if SYNONYM is set OFF. (See “SET SYNONYM” on page 351).
2. The subcommand name SET is required with this option (to avoid conflict with the MACRO subcommand).

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET MASK

```
►►──┬─────┬──MASK──┬─Define──────────────┬──────────────────────────────────────────►◄
    └─SET─┘        │             (1)     │
                   │          ┌───────┐  │
                   ├─Immed────┴───────┴──┤
                   │          └─text──┘  │
                   └─Modify──────────────┘
```

Notes:

1 If *text* is not specified, it replaces the current mask with a blank line.

Purpose

Whenever a subcommand or macro inserts a new line in a file, it is prefilled with the contents of a mask. Initially, the mask contains all blanks. You can use the MASK option to change the contents of the mask.

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

Operands

Define
after enter is pressed, displays the scale in the command line; you can type over it to define a new mask. The width of the screen limits the size of the mask. (If you do not type a new mask over the scale, the scale becomes the new mask.)

Immed
Immed *text*
replaces immediately the current mask with the specified text. If *text* is not specified, it replaces the current mask with a blank line.

Modify
after enter is pressed, displays the current mask in the command line. You can *type over the mask* to redefine it. (The scale does not appear to assist you in defining the mask, as it does with the DEFINE operand.)

Initial Setting

The initial setting of the mask is a blank line.

Usage Notes

1. All subcommands and macros that insert lines in a file prefill the new line with the mask. These subcommands are ADD, INPUT, and REPLACE, and the prefix subcommands A and I. The macros are SI and the prefix macro SI.
2. You can use the QUERY MASK subcommand to display the current mask in the message line, without changing it.
3. When using SET MASK DEFINE or SET MASK MODIFY to reset the mask to a blank line, you can press the spacebar once and then press the ERASE EOF key; this prevents you from clearing the mask if you accidentally press ERASE EOF. Pressing only ERASE EOF does not clear the mask.

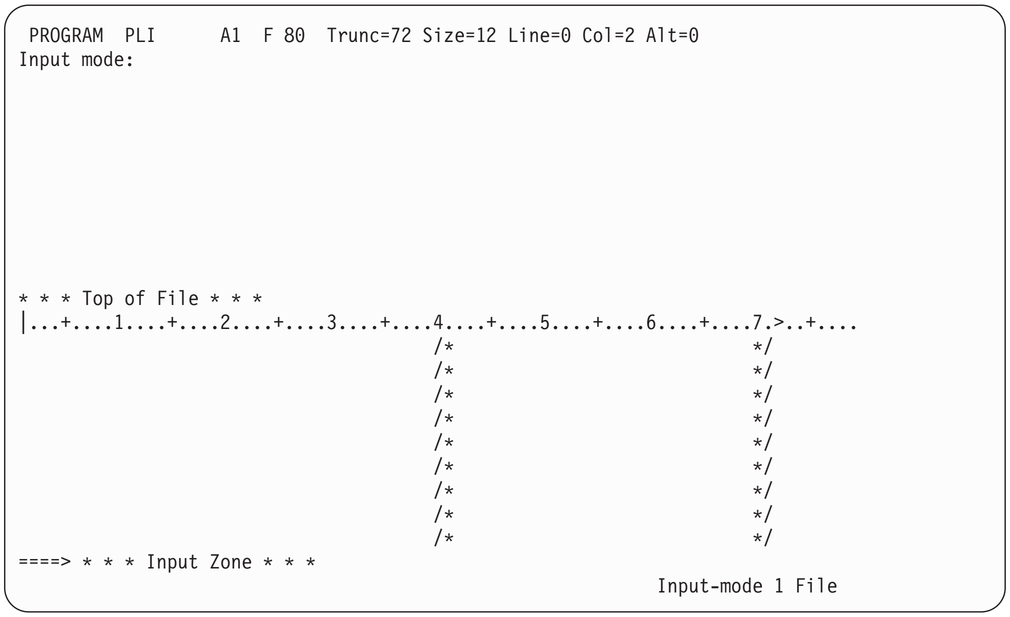
Notes for Macro Writers

1. The EXTRACT/MASK/ subcommand returns the current mask.

Examples

Figure 13 is an example of using SET MASK to define the “comments” area in a PLI program.

1. Enter the following subcommand:
```
====> set mask define
```
2. The scale appears in the command line:
```
====> ....+....1....+....2....+....3....+....4....+....5....+....6....+....7...
+....8....+....9....+...10....+...11....+...12....+...13... X E D I T  1 File
```
3. Type over the scale to define the mask, deleting any scale characters, then press enter.
```
====>                                       /*                            */
                                                            X E D I T  1 File
```
4. Enter the INPUT subcommand. Each line in the input zone contains the mask:



```
 PROGRAM  PLI      A1  F 80  Trunc=72 Size=12 Line=0 Col=2 Alt=0
Input mode:









* * * Top of File * * *
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7.>..+....
                                       /*                            */
                                       /*                            */
                                       /*                            */
                                       /*                            */
                                       /*                            */
                                       /*                            */
                                       /*                            */
                                       /*                            */
                                       /*                            */
====> * * * Input Zone * * *
                                                            Input-mode 1 File
```

Figure 13. Using the SET MASK Subcommand

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET MSGLINE

```
                                                  ┌─1─┐
►►──┬─────┬─MSGLine──┬─ON──┬─M───┬──┼───┼──┬─────────┬─┬────────────────────────────────►◄
    └─SET─┘          │     ├─Mn──┤  └─p─┘  └─Overlay─┘ │
                     │     ├─ M+n┤                     │
                     │     ├─ M−n┤                     │
                     │     ├─n───┤                     │
                     │     ├─ +n ┤                     │
                     │     └─ −n ┘                     │
                     └─OFF─────────────────────────────┘
```

Purpose

Use the MSGLINE option to define the location of the message line on the screen, and the maximum number of lines a message can occupy. It can also be used to determine whether a blank line is usually displayed for the message line.

Operands

ON
allows you to receive messages at the screen location defined.

OFF
turns off the message line. All XEDIT messages are passed to CMS. See Usage Note 1 on page 290 for information on displaying messages passed to CMS.

M stands for the middle of the screen (rounded up for odd-sized screens).

M*n*
M+*n*
M–*n*
The M can be combined with a constant positive (+ is implicit) or negative integer to mean *n* lines below the middle of the screen (M+*n*) or *n* lines above the middle of the screen (M–*n*).

n
+*n*
–*n* indicates the MSGLINE is located *n* lines from the top of the screen (+ is implied or specified) or from the bottom of the screen (–*n*). The default is +.

p is the maximum number of lines that can be used to display messages. The number of lines to display messages does not exceed the number of lines in the actual messages. If the messages do not fit on *p* lines, the messages are passed to CMS. See Usage Note 1 on page 290 for information on displaying messages passed to CMS.

Overlay
indicates you do not want a blank line on the screen for messages. If OVERLAY is specified, no message line is displayed unless a message is issued. When OVERLAY is specified, reserved lines (including the scale, tabline, and so forth) and file lines are displayed wherever the message line was defined until a message is issued. If OVERLAY is not specified, a blank message line is displayed wherever the message line was defined. If the message line is defined as the same line as the command line, the command line overlays the blank MSGLINE so a command line is available.

Initial Setting

```
MSGLINE ON 2 2
```

Usage Notes

1. If multiple messages that do not fit on the message line(s) need to be displayed, or MSGLINE is set OFF, messages are passed to CMS to be displayed.

   When full-screen CMS is ON, the CMSOUT window connected to the CMS virtual screen contains the output. (Output appears in the CMSOUT window by default; you can route the output to another window by using the CMS VSCREEN ROUTE command, explained in *z/VM: CMS Commands and Utilities Reference*.) To see all of the information in the virtual screen, you can use the CMS WINDOW FORWARD or WINDOW BACKWARD command. The screen is cleared automatically when you scroll to the bottom of the file. Alternately, you can clear the screen with the CMS WINDOW DROP command. If you delete the CMSOUT window, you do not see messages passed to CMS.

   When full-screen CMS is OFF, the screen is cleared to display the message(s). Press CLEAR to redisplay the file. No alarm sounds on either the CMS screen or the next XEDIT screen.

   If you are in a GUI environment, messages that do not fit in the message area are displayed in the XEDIT Messages window.
2. The command line and MSGLINE may be the same line. If the messages overlay the command line you can press CLEAR (or any key that does not produce a message) to restore the command line.
3. When full-screen CMS is ON and messages are passed to CMS, the messages are displayed with the XEDIT msgline color attributes — color, highlighting, programmed symbol set, and so forth.
4. If a message line occupies more than one screen line, 256 bytes (the maximum length), will be displayed without trailing blanks.
5. For more information on the effects of the SET MSGLINE subcommand on wrapped lines, see page 368.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
521E	**Invalid line number [RC=5]**
526E	**Option** *option* **valid in display mode only [RC=3]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
3	Subcommand valid only for display terminal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET MSGMODE

```
                                      (1)
                                   ┌────────┐
►►──┬─────┬──MSGMode──┬─ON──┬──────┼────────┼──────────────────────────────────────────────────►◄
    └─SET─┘           └─OFF─┘      ├─Short──┤
                                   └─Long───┘
```

Notes:

1 The default is the previous setting.

Purpose

Use the MSGMODE option to control the message display.

Operands

ON
displays all messages at the terminal.

OFF
messages and data are not displayed.

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.

Long
displays all messages in their full length.

Initial Setting

`MSGMODE ON LONG`
(unless the NOMSG option was specified on the XEDIT command).

Usage Notes

1. When MSGMODE is set to OFF, messages are not displayed but are still generated internally (in long or short form, according to the setting). However, you can display the last message generated by using the following sequence:
   ```
   set msgmode on
   query lastmsg
   ```
   In a macro, EXTRACT/LASTMSG/ can get the last message generated.
2. To enter XEDIT with the initial setting of MSGMODE OFF, use the NOMSG option on the XEDIT or LOAD subcommand. See Chapter 2, "The XEDIT Command," on page 11 for an explanation of the NOMSG option.
3. The MSGMODE setting is handled differently for the message COUNT issues containing the number of occurrences of a string. See "COUNT" on page 68 for more information.
4. When a macro is running, some messages (mostly informational) are not issued.

Examples

In the following example, SET MSGMODE displays all messages.

```
set msgmode on
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET NAMETYPE

```
►►──┬─────┬──NAMetype──┬─CMS─┬──────────────────────────────────────────────►◄
    └─SET─┘            └─BFS─┘
```

Purpose

Use the NAMETYPE option to specify whether the file IDs entered on subcommands will be interpreted to be in the form used for CMS record files or in the form used for byte file system (BFS) files.

SET NAMETYPE will change the format of the displayed file ID to either the CMS or BFS format. If the file ID has not been properly initialized through the use of the SET FNAME, SET FTYPE, and SET FMODE subcommands (for an SFS or minidisk file) or the SET PNAME subcommand (for a BFS file), blanks will be displayed.

Operands

CMS
indicates file IDs entered on subcommands (other than the XEDIT subcommand itself) will be interpreted to be CMS record files (*fn ft fm*).

BFS
indicates file IDs entered on subcommands (other than the XEDIT subcommand itself) will be interpreted to be in the BFS format (*pathname*). See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name syntax.

Initial Setting

The initial setting for SFS and minidisk files is CMS.

The initial setting for BFS files is BFS.

Usage Notes

1. When NAMETYPE BFS is in effect, the path name is converted to uppercase only when SET CASE UPPER is in effect.
2. The NAMETYPE setting applies to the following subcommands when a file identifier or equal sign (=) is specified:
   - FILE
   - GET
   - PUT
   - PUTD
   - SAVE

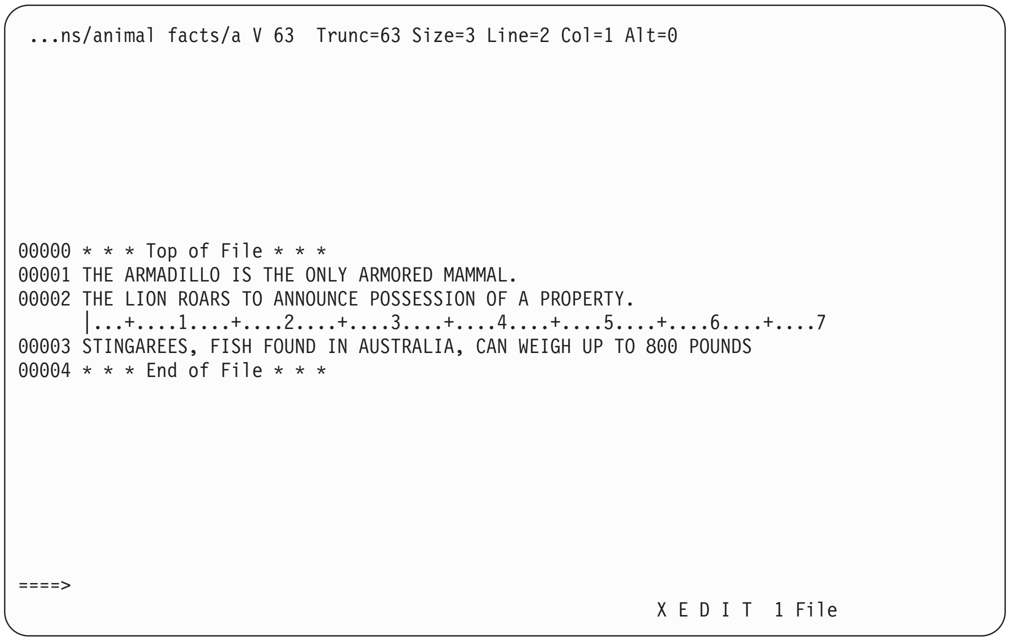
   There is also a NAMETYPE option on the LOAD and XEDIT commands.

Examples

The following XEDIT screen was created by entering:

```
XEDIT 'childrens/animal facts/a' (NAMETYPE BFS
```

Up to 20 characters of the file ID (*pathname*) are displayed. If there are more than 20 characters in the path name, the end of the name is displayed and '...' indicates part of the name is not displayed. Use the QUERY PNAME subcommand to display the entire file ID.

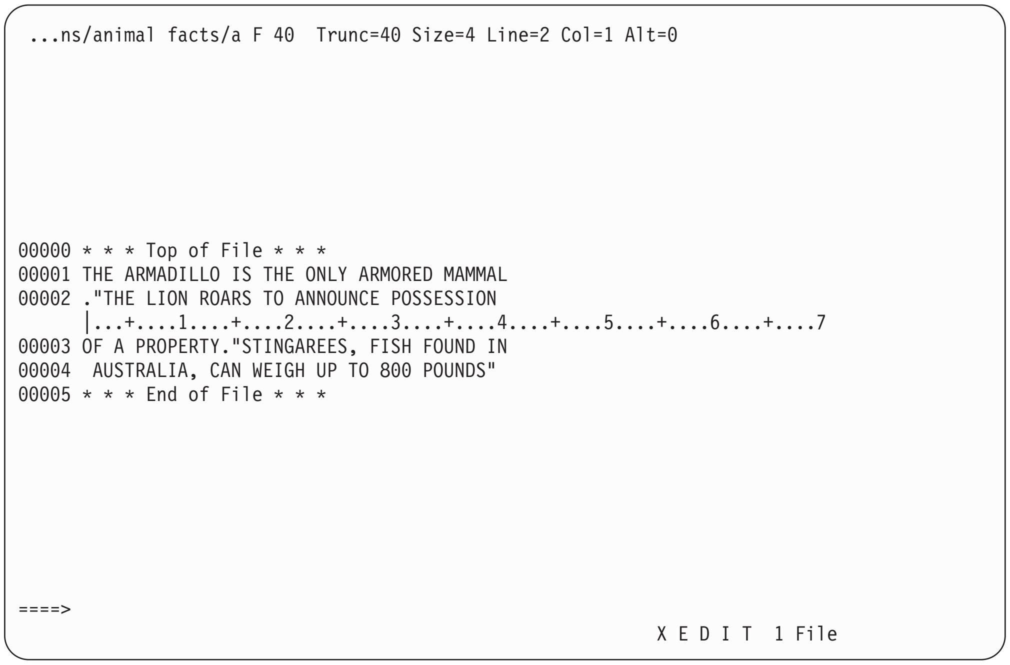


```
 ...ns/animal facts/a V 63  Trunc=63 Size=3 Line=2 Col=1 Alt=0








00000 * * * Top of File * * *
00001 THE ARMADILLO IS THE ONLY ARMORED MAMMAL.
00002 THE LION ROARS TO ANNOUNCE POSSESSION OF A PROPERTY.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
00003 STINGAREES, FISH FOUND IN AUSTRALIA, CAN WEIGH UP TO 800 POUNDS
00004 * * * End of File * * *








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

Figure 14. Sample XEDIT Screen of BFS file.

If you enter QUIT, you can edit the same file in a different manner by entering:

```
XEDIT 'childrens/animal facts/a' (NAMETYPE BFS BFSLINE 40
```



```
 ...ns/animal facts/a F 40  Trunc=40 Size=4 Line=2 Col=1 Alt=0








00000 * * * Top of File * * *
00001 THE ARMADILLO IS THE ONLY ARMORED MAMMAL
00002 ."THE LION ROARS TO ANNOUNCE POSSESSION
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
00003 OF A PROPERTY."STINGAREES, FISH FOUND IN
00004  AUSTRALIA, CAN WEIGH UP TO 800 POUNDS"
00005 * * * End of File * * *








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

Figure 15. Sample XEDIT Screen of BFS file with different BFSLINE settings.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
554E	**Not enough virtual storage available [RC=104]**

where return codes are:

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

SET NONDISP

```
                      (1)
►►──┬─────┬──NONDisp──┬──────┬────────────────────────────────────────────────►◄
    └─SET─┘           └─char─┘
```

Notes:

1 If *char* is not specified, a blank is used.

Purpose

Use the NONDISP option to define for XEDIT and CMS a character to be used in place of nondisplayable characters.

Operands

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

Initial Setting

Defined by the CMS setting.

Usage Notes

1. You can specify the nondisplayable character in hexadecimal if SET HEX ON is in effect.
2. Setting a nondisplayable character to a visible one (for example, SET NONDISP ") is helpful when doing full-screen changes on display terminals. When changing lines on the display, XEDIT tries to preserve nondisplayable characters within modified lines by comparing the line returned from the terminal with its old contents. This may lead to unexpected changes, particularly if you use the DELETE or INSERT keys.
3. The NONDISP character (*char*) you specify must be a displayable character. Nondisplayable characters are ignored.
4. If you issue CMS SET NONDISP while in XEDIT, the new nondisplayable character does not appear in a line containing nondisplayable characters until you alter the line. Similarly, if you issue CMS SET OUTPUT while in XEDIT, the changed character does not appear until you alter the line containing the character(s) to be changed.
5. The translation of the nondisplayable character depends on the type of terminal and whether SET APL ON or SET TEXT ON is in effect. For more information on the nondisplayable character translation tables, see *z/VM: CMS Application Development Guide*.
6. Changing the editor NONDISP setting also changes the CMS NONDISP setting.
7. If you are in a GUI environment, SET NONDISP is ignored.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET NULLS

```
►►──┬─────┬──NULls──┬─ON──┬──────────────────────────────────────────────►◄
    └─SET─┘         └─OFF─┘
```

Purpose

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').

Operands

ON
specifies 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 trailing blanks are not to be replaced with nulls. The insert key cannot be used to insert characters in a line unless the ERASE EOF key removes at least as many trailing blanks as characters to be inserted.

Initial Setting

```
NULLS  OFF
```

Usage Notes

1. You can use any key defined as *NULLKEY* instead of issuing SET NULLS ON if you wish to use the insert key for only one line. The *NULLKEY* function sets NULLS ON in the field that contains the cursor. If the cursor is on a prefix area, blanks are changed to nulls on the file line associated with that prefix area. If you move the cursor to another field and wish to use insert mode, press the key defined as *NULLKEY* again. The editor initially assigns *NULLKEY* to the PA2 key.
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.
3. See “SET FULLREAD” on page 275.
4. If you are in a GUI environment, SET NULLS ON is recommended and will be in effect during INPUT mode.

Examples

For more information, see *z/VM: XEDIT User’s Guide*.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET NUMBER

```
►►──┬─────┬──NUMber──┬─ON──┬────────────────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

Use the NUMBER option to specify whether file line numbers are to be displayed in the prefix area.

Operands

ON
displays a five-digit line number in the prefix area of the lines. The line numbers are sequential and are recomputed when lines are added or deleted.

OFF
displays five equal signs (=====) in the prefix area of each line, unless the prefix area has been set to NULLS.

Initial Setting

```
NUMBER  OFF
```

Usage Notes

1. In some GUI environments, it may be helpful to issue SET NUMBER OFF and SET PREFIX NULLS to ease typing in the prefix area. Alternatively, you may need to do one of the following:
   - toggle the insert key or
   - delete characters.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET PAn

```
                          (1)        (2)
                        ┌───────┐  ┌───────┐
►►──┬─────┬──PAn────────┼───────┼──┼───────┼──────────────────────────────────────────►◄
    └─SET─┘             ├─BEFORE┤  ├─string┤
                        ├─AFTER─┤  ├─NULLKEY┤
                        ├─ONLY──┤  ├─COPYKEY┤
                        └─IGNORE┘  └─TABKEY─┘
```

Notes:

1 Refer to the BEFORE operand description for the default.

2 If *string* is not specified, the PA key is undefined.

Purpose

Use the PA*n* option to define a meaning for a specified hardware attention (PA) key or to remove the meaning associated with the specified PA key.

Operands

n specifies a PA key number (1, 2, or 3). Issuing SET PA*n* (without an additional operand) removes the meaning associated with that PA key.

BEFORE
specifies the key definition is executed before the contents of the command line. BEFORE is the default for all the keys, unless the string begins with ? or =. For these two strings, ONLY is the default.

AFTER
specifies the key definition is executed after the contents of the command line.

ONLY
specifies only the key definition is executed, thereby ignoring the command line.

IGNORE
specifies the key definition is ignored when you enter something on the command line, thereby executing only the command line.

string
is any XEDIT subcommand or macro (including CP, CMS, or EXEC) executed when you press the PA key. If *string* is null, the PA key is undefined.

NULLKEY
When you press the corresponding PA key, blank characters are changed to nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, blanks are changed to nulls on the file line associated with that prefix area.

TABKEY
When you press the corresponding PA key, the cursor is moved into the next tab column, as defined in the SET TABS subcommand.

COPYKEY
When you press the corresponding PA key, the exact content of the virtual screen is copied into the printer spool.

Initial Setting

```
SET PA1   BEFORE COMMAND CMS WINDOW POP WM
SET PA2   BEFORE NULLKEY
SET PA3   ONLY ?
```

Usage Notes

1. To assign a sequence of subcommands to a single PA key: set off the LINEND character; set the PA key to the subcommands, separating subcommands with LINEND characters; set the LINEND character back on before using the PA key.

   For example:

   ```
   set linend off
   set pa2 next#c/A/B
   set linend on #
   ```
2. If you press a PA key set to TABKEY and no tab columns are available on the screen, the position of the cursor remains unchanged.
3. PA keys can be used in input mode. (See “Usage Notes” on page 132.)
4. When you use a PA key set to COPYKEY, you should close the printer at the end of the editing session. If an error occurs in printing the screen, no copy is placed in the printer and only severe error messages are issued.
5. When you press a PA key set to TABKEY, COPYKEY, or NULLKEY, no screen changes are processed including the command line, and the keywords BEFORE, AFTER, ONLY, and IGNORE have no effect.
6. Setting FULLREAD ON prevents you from losing any screen changes when you press a PA key and a message is displayed on a cleared screen.
7. Setting the CP break key (PA1 by default) overrides any later defined editor setting unless SET BRKKEY OFF is specified.
8. When the PA key is pressed, the PA key definition, the contents of the command line, or both are placed on the console stack in the order described in “Using READ CMDLINE” on page 205. Thus if BEFORE is specified, execution of the PA key definition may alter or suppress execution of the command line by altering the contents of the stack; similarly, if AFTER is specified, execution of the command line may alter or suppress execution of the PA key definition.
9. If you are in a GUI environment, *n* NULLKEY is ignored.

Messages and Return Codes

520E **Invalid operand:** *operand* **[RC=5]**
525E **Invalid {PFkey|PFkey/PAkey} number [RC=5]**
545E **Missing operand(s) [RC=5]**
614E **Screen modifications lost. See SET FULLREAD to use PAkeys safely. [RC=8]**
657E **Undefined PFkey/PAkey**

where return codes are:

0	Normal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Modifications lost because PA key pressed when message pending

SET PACK

```
►►──┬─────┬──PACK──┬─ON──┬──────────────────────────────────────────────────►◄
    └─SET─┘        └─OFF─┘
```

Purpose

Use the PACK option to specify whether the editor is to pack the file when it is written to disk or Shared File System (SFS) directory.

Operands

ON
specifies the editor is to compress records in a file so they can be stored in packed format when a FILE or SAVE subcommand is issued.

OFF
specifies the editor is not to pack the file when it is written to disk or directory.

Initial Setting

Based on 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 you issue an XEDIT command for a file in packed format on disk or directory, the editor automatically issues a SET PACK ON subcommand. Thus, when the file is filed or saved, it is written back to disk or directory in packed format.
2. The PACK option is compatible with the PACK option of the CMS COPYFILE command. You can pack (or unpack) a file with either the editor SET PACK subcommand or the CMS COPYFILE command.
3. Do not modify a file in packed format in any way while it is on disk or directory (for example, a PUT subcommand would append data to a file while it is on disk or directory). If you modify such a file, neither XEDIT nor the CMS COPYFILE command can reconstruct it.
4. SET PACK is ignored for byte file system (BFS) files.

Responses

The editor displays the record format in the IDLINE as:

```
FP
```

(fixed packed)

or

```
VP
```

(variable packed)

Messages and Return Codes

520E Invalid operand: ***operand*** **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET PENDING

```
►►──┬─────┬──PENDing──┬─┬─ON────┬──string─┬──────────────────────────────────►◄
    └─SET─┘           │ ├─BLOCK─┤         │
                      │ └─ERROR─┘         │
                      └─OFF───────────────┘
```

Purpose

Use the PENDING option to control the execution of a prefix macro and the status of the screen while the prefix macro is being executed. This option, designed to be issued from a macro, can display a prefix subcommand or macro in the prefix area of the current line, or indicate to the user a prefix subcommand or macro that was entered is the beginning or end of a block. In both cases, the screen is placed in a pending status (described in the following notes). SET PENDING can also be used to notify the user a prefix macro was entered incorrectly.

Operands

ON *string*
displays *string*, which can be any prefix subcommand or macro, in the prefix area of the current line and displays a pending notice in the status area (described in the following notes). By default, the *string* in the prefix area is highlighted. This form of SET PENDING adds an entry to the *pending list* (list of pending prefix subcommands and macros), which is described in the notes following.

BLOCK *string*
displays *string,* which is the block form of any prefix subcommand or macro (for example, DD is the block form of the D prefix subcommand), in the prefix area of the current line where it was entered and displays a pending notice in the status area (described in the notes following). By default, the *string* in the prefix area is highlighted. This form of SET PENDING notifies the user that the beginning (or end) of a block was entered. The *string* would usually be the same as that entered by the user, which is determined by the macro. This form of SET PENDING adds an entry to the *pending list,* which is described in the notes following.

ERROR *string*
displays *string* in the prefix area of the current line, prefixed by a question mark (?). By default, the *string* in the prefix area is highlighted. This indicates an error occurred while a prefix macro (or subcommand) was executing, for example, if the macro was entered incorrectly. The *string* would normally be the incorrectly-entered prefix macro, as determined by the prefix macro, so the user could correct the error.

Because this entry in the pending list is deleted the next time the list is processed, pressing ENTER while `? string` is displayed resets the prefix area. (This prevents subsequent attempts by the user to execute an incorrect prefix subcommand or macro.) This form of SET PENDING does not display a pending notice.

OFF
removes a pending prefix subcommand or macro from the prefix area of the current line (that is, removes this entry from the pending list), and resets the prefix area and the status area.

Initial Setting

The pending list is empty.

Notes for Macro Writers

1. The *pending list* is a list of prefix subcommands and macros that have not yet been executed. Every time the editor reads the screen, the pending list is updated with any new prefix subcommands and macros, each of which adds an entry to the list. Each entry is associated with a specific line in the file. The pending list is executed when it is updated. If a prefix macro returns a nonzero return code, execution of the pending list stops and all entries not executed remain pending, until the user presses a key.

   An entry is deleted from the pending list when it is executed or overtyped on the user's screen with a new prefix subcommand, prefix macro, or blanks. An entry can also be added by using SET PENDING ON *string* and deleted by issuing SET PENDING OFF or the RESET subcommand.

   A prefix macro can control its execution and the screen status by examining information in the pending list (using EXTRACT /PENDING. . ./) and by examining the *argument string* that is automatically passed to a prefix macro when it is invoked. The argument string contains information pertinent to the prefix macro when it is called, for example, the file line of the prefix area which it was entered. For more information on the description of this argument string, see *z/VM: XEDIT User's Guide*. The macro can then take appropriate action, which may include using some form of SET PENDING.
2. On display terminals, the following notice in the status area indicates the pending status: `value pending...` where *value* is the name of the pending prefix macro or subcommand entered in the prefix area. If multiple prefix subcommands or macros are pending, the first one (starting from the top of file) is displayed in the pending notice.
3. After all the prefix subcommands and macros in the pending list are processed, the current line is restored to what it was before processing. Therefore, a prefix macro does not usually need to be concerned with restoring the current line.

   To set the current line, you must override the automatic current line restoration by issuing the command:

   ```
   set pending on /
   ```

   When multiple / prefix subcommands are entered, the last one executed sets the current line.
4. When a macro or subcommand is first put in the pending list, it is checked (with the CMS STATE command) to see if it exists. If it does not exist, it is left pending. Creating the macro does not cause the pending macro to be automatically executed. Reenter it into the prefix area for it to be executed.
5. When the pending list is executed, it is scanned for SCALE, TABL, and / prefix subcommands after all the other prefix subcommand and macros have been executed. SCALE, TABL, and / are executed in the logical screen in which they were issued (not just on the screen in which the pending list was executed). Because they are executed after the rest of the pending list is processed, a prefix macro can specify them and they can be executed before the screen is redisplayed (this is particularly useful for setting the current line from a prefix

macro). When multiple SCALE, TABL, and / prefix subcommands are issued, only the last one issued is used. If the last TABL or SCALE is specified on a line that does not fall on the screen, it is ignored.

6. For more information on the pending list and writing prefix macros, see the subcommands EXTRACT/QUERY PENDING, and SET/EXTRACT/QUERY PREFIX SYNONYM. For information on IBM-supplied prefix macros, see Chapter 4, "Prefix Subcommands and Macros," on page 419. For tutorial information on writing prefix macros and examples of how to use SET PENDING in prefix macros, see *z/VM: XEDIT User's Guide*.

Examples

This section shows examples of the SET PENDING subcommand.

Example 1

In the following example, entering

```
set pending on A
```

adds an entry to the pending list and displays `A====` in the prefix area of the current line and the notice `A pending. . .` in the status area.

Example 2

In the following example, entering,

```
:3 set pending off
```

removes the current line from the pending list, that is, makes line 3 the current line and resets the prefix area and the status area.

Example 3

In the following example, entering

```
:10 set pending error XX20
```

makes line 10 the current line and displays `?XX20` in the prefix area. (`XX20` is an invalid form of the X prefix macro.)

Example 4

In this example, entering

```
:100 set pending block XX
```

makes line 100 the current line and displays `XX` in the prefix area (highlighted), and displays `XX pending. . .` in the status area.

Assuming this was issued from a prefix macro, note that :100 is an absolute line number target. It makes this line current for the SET PENDING subcommand, which operates on the current line. Because the current line is restored after the pending list is executed (see notes preceding), the block form of the macro is displayed in the prefix area of the line which was current when SET PENDING was issued (which is not necessarily the current line when the screen is displayed).

Therefore, if XX is entered on line 100 of the file, and line 100 is not the current line, :100 makes line 100 current for the SET PENDING subcommand. However, because the current line is restored after execution, the user sees `XX` displayed in line 100, which was the current line at the time SET PENDING was issued.

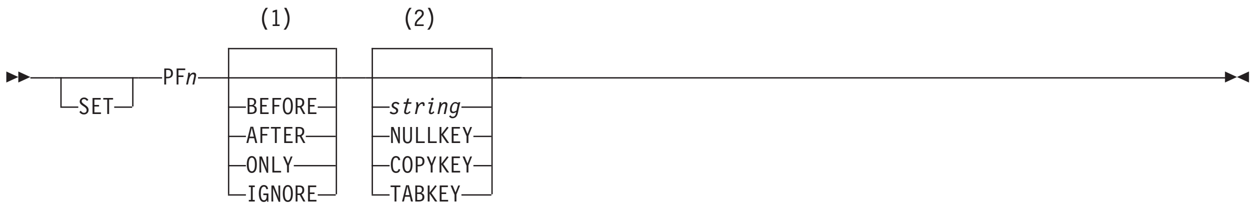
Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET PFn



Notes:

1 Refer to the BEFORE operand description for the default.

2 If the *string* is not specified, the PF key is undefined.

Purpose

Use the PF*n* option to define a meaning for a specified hardware program function (PF) key or to remove the meaning, if any, associated with the specified PF key.

Operands

n specifies a PF key number from 1 - 24. To remove the meaning associated with a PF key, enter SET PF*n* with no additional operand.

BEFORE
specifies executing the key definition before the contents of the command line. BEFORE is the default for all the keys, unless the *string* begins with a ? or =. For these two strings, ONLY is the default.

AFTER
specifies executing the key definition after the contents of the command line.

ONLY
specifies executing only the key definition, thereby ignoring the command line.

IGNORE
specifies ignoring the key definition when you enter something on the command line, thereby executing only the command line.

string
is any XEDIT subcommand or macro (including CP, CMS, or EXEC) executed when you press the PF key. If *string* is null, the PF key is undefined.

NULLKEY
When you press the corresponding PF key, blank characters are changed to nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, blanks are changed to nulls on the file line associated with that prefix area.

TABKEY
When you press the corresponding PF key, the cursor is moved into the next tab column, as defined in the SET TABS subcommand.

COPYKEY
When you press the corresponding PF key, the exact content of the virtual screen is copied into the printer spool.

Initial Setting

```
SET PF01 BEFORE HELP MENU
SET PF02 BEFORE SOS LINEADD
SET PF03 BEFORE QUIT
SET PF04 BEFORE TABKEY
SET PF05 BEFORE SCHANGE 6
SET PF06 ONLY   ?
SET PF07 BEFORE BACKWARD
SET PF08 BEFORE FORWARD
SET PF09 ONLY   =
SET PF10 BEFORE RGTLEFT
SET PF11 BEFORE SPLTJOIN
SET PF12 BEFORE CURSOR HOME
```

Note: These are the initial settings. On a terminal equipped with 24 PF keys, PF keys 13 through 24 have the same values as PF keys 1 through 12 (described previously), except PF key 17 is set to BEFORE SCHANGE 18.

Usage Notes

1. To assign a sequence of subcommands to a single PF key: set off the LINEND character; set the PF key to the subcommands, separating the subcommands with LINEND characters; set the LINEND character back on before using the PF key.

   For example:

   ```
   set linend off
   set pf3 next#c/A/B
   set linend on #
   ```
2. If you press a PF key set to TABKEY and no tab columns are available on the screen, the position of the cursor remains unchanged.
3. To get the same effect as a CP SET PF DELAY, use:

   ```
   set pfn cmsg...
   ```
4. You can use PF keys in input mode. (See “Usage Notes” on page 132.)
5. When you use a PF key set to COPYKEY, you should close the printer at the end of the editing session. If an error occurs in printing the screen, no copy is placed in the printer and only severe error messages are issued.
6. When you press a PF key set to TABKEY, COPYKEY or NULLKEY, no screen changes are processed, including the command line, and the keywords BEFORE, AFTER, ONLY and IGNORE have no effect.
7. When the PF key is pressed, the PF key definition, the contents of the command line, or both are placed on the console stack in the order described in “Using READ CMDLINE” on page 205. Thus if BEFORE is specified, execution of the PF key definition may alter or suppress execution of the command line by altering the contents of the stack; similarly, if AFTER is specified, execution of the command line may alter or suppress execution of the PF key definition.
8. If you are in a GUI environment, NULLKEY is ignored. Also, settings for PF keys PF13 through PF24 are executed by holding down the Shift key and pressing PF1 through PF12. The Shift and PF key combinations correspond as follows:

   Shift + PF01 = PF13
   Shift + PF02 = PF14
   Shift + PF03 = PF15
   Shift + PF04 = PF16
   Shift + PF05 = PF17
   Shift + PF06 = PF18

Shift	+	PF07	=	PF19
Shift	+	PF08	=	PF20
Shift	+	PF09	=	PF21
Shift	+	PF10	=	PF22
Shift	+	PF11	=	PF23
Shift	+	PF12	=	PF24

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
525E Invalid {PFkey|PFkey/PAkey} number [RC=5]
545E Missing operand(s) [RC=5]
657E Undefined PFkey/PAkey

where return codes are:

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

SET PNAME

```
►►──┬─────┬──PName──┬────────┬──pathname_fragment──────────────────────────────►◄
    └─SET─┘         └─APpend─┘
```

Purpose

Use the PNAME option to change the name of the byte file system (BFS) file being edited.

Operands

APpend
specifies the *pathname_fragment* is to be appended to the end of the BFS path name in effect for the current editing session, forming a new BFS path name. If APPEND is not specified, the path name of the file being edited is changed to *pathname_fragment*.

pathname_fragment
is the BFS path name or a portion of the new path name of the BFS file being edited. The length of a path name entered on subcommands is limited. If you wish to set a long path name, enter `PN` *`pathname_fragment1`* and then enter `PN AP` *`pathname_fragment2`* one or more times. See "Understanding Byte File System (BFS) Path Name Syntax" on page 4 for a description of the different forms of the BFS path name syntax.

Initial Setting

PNAME is the path name of the BFS file being edited, or null if the file being edited is an SFS or minidisk file.

Usage Notes

1. The specified path name is used the next time a FILE or SAVE is entered without operands when NAMETYPE BFS is in effect. If the file being edited was written to the BFS earlier, that copy of the file remains unchanged.
2. If an object already exists with the resulting path name and you enter FILE or SAVE, the editor displays an error message.
3. If you are editing an SFS or minidisk file, the initial setting for PNAME is all nulls. SET PNAME is allowed if you are editing an SFS or minidisk file. However, it is not used until SET NAMETYPE BFS is in effect.
4. SET PNAME will change the portion of the path name displayed in the file ID field of the XEDIT session if NAMETYPE BFS is in effect.
5. Path name can be specified in several ways:
   - When there is no ***/*** at the start of the file identifier, the current working directory will be assumed to prefix the specified file or directory name. This form of path name is called a **relative path name**. Refer to OPENVM SET DIRECTORY and OPENVM QUERY DIRECTORY in *z/VM: OpenExtensions Commands Reference* to change and find the value of the current working directory.
   - If the path name is specified as */myname*, the root directory will be assumed to prefix the specified path name. This form of path name is called an

absolute path name. See OPENVM MOUNT and OPENVM QUERY MOUNT in *z/VM: OpenExtensions Commands Reference* to change and find the value of the root directory.

- When `/../VMBFS:`*`filepoolid`*`:`*`filespaceid`* is specified at the start of a path name, the path name will be referred to as being **fully qualified**.

Messages and Return Codes

054E	**Incomplete [or incorrect] fileid [RC=24]**
062E	**Invalid character in fileid** *operand* **[RC=20]**
520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
554E	**Not enough virtual storage available [RC=104]**
555E	**File** *pathname* **already in storage [RC=4]**

where return codes are:

0	Normal
4	File is already in storage
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been entered in a macro called from the last file in the ring
20	Invalid character in BFS path name
24	Path name or path name fragment not valid
104	Insufficient storage available

SET POINT

```
►►──┬─────┬──Point──.symbol──┬─────┬──────────────────────────────────────────►◄
    └─SET─┘                  └─OFF─┘
```

Purpose

Use the POINT option to define a symbolic name for the current line. You can define more than one name for a line by issuing separate SET POINT subcommands. You can use these names to refer to the line in subsequent target operands of XEDIT subcommands.

Operands

.*symbol*
: is a symbolic name for the current line. The name must begin with a period and be followed by 1 - 8 alphanumeric or special characters, for example, .AAA.

OFF
: deletes the specified *symbol*, without moving the line pointer. The *symbol* to be deleted must be specified before this operand.

Initial Setting

No names are initially defined.

Usage Notes

1. The POINT option makes it unnecessary for you to remember or to look up the line number of a line. You can reference a line by its name at any time during the editing session. For example, if you enter the following subcommand:

   ```
   set point .XAVIER
   ```

   the current line is assigned the specified name.

   You can then reference the name at any time during the editing session. For example:

   ```
   move 3 .XAVIER
   ```

   moves three lines, beginning with the current line, after the line named .XAVIER.
2. The line number of a line can change during an editing session; for example, adding lines before a particular line increments its line number. The symbolic name, however, stays with a line for the entire editing session.
3. The *.xxxx* prefix subcommand can also assign a symbolic name to a line. In this case, the symbolic name is limited to four characters.

   You can delete a symbolic name for a line by using *.xxxx* to assign that name to another line.
4. You can use the QUERY POINT subcommand to display the symbolic name(s) of the current line. You can use QUERY POINT * to display all symbolic names and their line numbers.

Notes for Macro Writers

1. The EXTRACT/POINT/ subcommand returns the symbolic name(s) and line number of the current line, and EXTRACT/POINT */ returns the symbolic names and their line numbers in the file.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
539E	**Named line not found [RC=2]**
540E	**Name already defined on line** *nn* **[RC=1]**
541E	**Invalid name [RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
1	Duplicate name defined
2	Name does not exist for OFF function
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET PREFIX

Format 1

```
                                 ┌─Left──┐
►►──┬─────┬──PREfix──┬─┬─ON────┬──┼───────┼─┬──────────────────────────────►◄
    └─SET─┘          │ └─Nulls─┘  └─Right─┘ │
                     └─OFF──────────────────┘
```

Format 2

```
►►──┬─────┬──PREfix──Synonym──newname──oldname──────────────────────────────►◄
    └─SET─┘
```

Purpose

The SET PREFIX subcommand has two formats. Use the first format to control the display of the prefix area. Use the second format to define a synonym for a prefix.

Operands

ON
displays a five-character prefix area (=====) for each logical line on the screen.

Left
Right
displays the prefix area on either the left (LEFT operand) or right (RIGHT operand) side of the screen. You can then enter prefix subcommands and macros in the prefix area.

OFF
removes the prefix area from the screen.

Nulls
displays nulls in the prefix area, enabling the insert key, and moves the prefix area to side of screen specified.

Synonym
is the keyword operand that indicates a synonym is to be defined for a prefix subcommand or macro.

newname
is a synonym to be assigned to a prefix subcommand or macro. The *newname* can be up to five alphabetic or special characters.

oldname
is the name of a prefix subcommand or macro for which you are defining a synonym. The *oldname* can be up to eight characters.

Initial Setting

```
PREFIX  ON  LEFT
```

Eight prefix synonyms are defined:

```
Newname      Oldname

   >         PRFSHIFT
  >>         PRFSHIFT
   <         PRFSHIFT
  <<         PRFSHIFT
   X         PREFIXX
  XX         PREFIXX
   S         PRFSHOW
.....        SI
```

Usage Notes

1. When using SET PREFIX ON with SET NUMBER ON, a five-digit line number is displayed instead of equal signs. When using SET PREFIX NULLS with SET NUMBER ON, any leading zeros are translated to nulls.
2. Using SET PREFIX OFF affects the number of columns used by XEDIT to display information.
3. In some GUI environments, it may be helpful to issue SET NUMBER OFF and SET PREFIX NULLS to ease typing in the prefix area. Alternatively, you may need to do one of the following:
   - toggle the insert key or
   - delete characters.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]
548E Invalid synonym operand: *operand* **[RC=5]**
554E Not enough virtual storage available [RC=104]

where return codes are:

0	Normal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
104	No storage available

SET RANGE

```
►►──┬─────┬──RANge──target1──target2──────────────────────────────────►◄
    └─SET─┘
```

Purpose

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.

Operands

target1
is the top of the range.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see “LOCATE” on page 147 and *z/VM: XEDIT User’s Guide*.

target2
is the bottom of the range.

Initial Setting

`RANGE 1` *n* where *n* is equal to the size of the file.

Usage Notes

1. After the range is defined, only the lines within the range are displayed. *Target1* becomes the equivalent of the first line in the file. Top of Range becomes the equivalent of the Top of File line. The TOP subcommand moves the line pointer to the line that precedes *target1* to allow insertion at the top of the range.

   Target2 becomes the equivalent of the last line in the file. End of Range becomes the equivalent of the End of File line. The BOTTOM subcommand moves the line pointer to *target2*.

   Note: Deleting all the lines in a range can result in an unexpected value for End of Range.
2. If you enter the SET RANGE option while the line pointer is outside the limits being defined, the first line of the new range becomes the current line.
3. FILE and SAVE subcommands write the *entire* file to disk or directory. No other subcommands have access to data outside the range.
4. To set a range outside the limits of the current range, specify *target1* and *target2* as absolute line numbers or a relative displacement from the current line.

   The following subcommand resets the range to the physical top and bottom of the file:

   ```
   set range -* *
   ```

   To set a range outside the limits of the current range, specify *target1* and *target2* as absolute line numbers. An alternate method is to issue:

   ```
   set range :1 *
   ```

(See Appendix B, “Effects of Selective Line Editing Subcommands,” on page 455 for information about SET RANGE and the SCOPE setting.)

5. SET RANGE cannot be issued when prefix subcommands or macros are pending and it cannot be issued from a prefix macro.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
528E	**Invalid range: target2 (line** *nn***) precedes target1 (line** *nn***) [RC=5]**
545E	**Missing operand(s) [RC=5]**
546E	**Target not found [RC=2]**
559W	**Warning: file is empty [RC=4]**
560E	**Not enough space for serialization between TRUNC and LRECL**
588E	**Prefix subcommand waiting... [RC=8]**

where return codes are:

0	Normal
2	Target not found
4	Range not set
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Prefix subcommand or macro waiting

SET RECFM

```
►►──┬─────┬──RECFm──┬─F──┬──────────────────────────────────────────────────►◄
    └─SET─┘         ├─V──┤
                    ├─FP─┤
                    └─VP─┘
```

Purpose

Use the RECFM option to define the record format for the file.

Operands

F specifies the record format is fixed.

V specifies the record format is variable.

FP specifies the record format is fixed packed.

VP specifies the record format is variable packed.

Initial Setting

- For SFS and minidisk files, the initial settings are based on the file type. See Appendix A, “File Type Defaults,” on page 453.
- For byte file system (BFS) files, the initial settings depend upon the BFSLINE value. For BFSLINE *lrecl*, the initial RECFM setting is fixed (F). For any other BFSLINE value, the initial RECFM setting is variable (V).

Usage Notes

1. When the record format is FP or VP, a SET PACK ON is automatically executed. For BFS files, SET PACK ON is ignored.
2. Changing the RECFM value affects how a BFS file is stored. When RECFM F is in effect, trailing blanks are not removed from the file when it is written out. When RECFM V is in effect, trailing blanks are removed.

Messages and Return Codes

515E RECFM must be F, V, FP, or VP [RC=5]
516E LRECL too large for V-format file [RC=4]
520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

0	Normal
4	Lrecl must be lower than 65 536 for recfm V when editing an SFS or minidisk file
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been entered in a macro called from the last file in the ring

SET REMOTE

```
►►──┬─────┬──REMOte──┬─ON──┬──────────────────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

Use the REMOTE option to control the way XEDIT and CMS handle the display for data transmission.

Operands

ON
specifies XEDIT and CMS are to compress the screen by removing nulls and combining data when five or more of the same characters occur consecutively in a data stream. This minimizes the amount of data transmitted and shortens the buffer, thus speeding transmission.

OFF
specifies XEDIT and CMS are not to compress the screen. Data is transmitted with no minimization.

Initial Setting

Defined by the CMS setting.

Usage Notes

1. Changing the editor REMOTE setting also changes the CMS REMOTE setting.
2. If you are in a GUI environment, SET REMOTE is ignored.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET RESERVED

```
►►──┬─────┬──RESERved──┬─M─────┬──┬─┤ Options ├─┬─────────────────────────────────────────────►◄
    └─SET─┘            ├─Mn────┤  └─Off─────────┘
                       ├─ M+n ─┤
                       ├─ M−n ─┤
                       ├─n─────┤
                       ├─ +n ──┤
                       └─ −n ──┘
```

Options:

```
   ┌─Default───┐  ┌─NONe─────┐  ┌─PS0─┐
├──┼───────────┼──┼──────────┼──┼─────┼──┬─High───┬──┬──────┬──────────────────────────────────┤
   ├─Blue──────┤  ├─BLInk────┤  ├─PSA─┤  └─Nohigh─┘  └─text─┘
   ├─Red───────┤  ├─REVvideo─┤  ├─PSB─┤
   ├─Pink──────┤  └─Underline┘  ├─PSC─┤
   ├─Green─────┤                ├─PSD─┤
   ├─Turquoise─┤                ├─PSE─┤
   ├─Yellow────┤                └─PSF─┘
   └─White─────┘
```

Purpose

Use the RESERVED option to reserve a specified line on the screen, thereby preventing the editor from using that line. The line can display blank or specified information with or without the following features:

- Color
- Extended highlighting
- Programmed symbol set
- Highlighting

The RESERVED option can also give a reserved line back to the editor. This option is designed to be issued from a macro.

Operands

M specifies the line that is to be the reserved line. The M stands for *middle of the screen* (rounded up for odd sized screens).

M*n*
M+*n*
M–*n*
You can combine M with a constant positive (+ is implicit) or negative integer to mean *n* lines below the middle of the screen (M+*n*), or *n* lines above the middle of the screen (M–*n*). For example, RESERVED M means the *middle of the screen*, and RESERVED M+3 means *three lines below the middle of the screen*.

n
+*n*
–*n* specifies the line that is to be the reserved line. The *n* or +*n* (+ is implicit) specifies the reserved line is displayed *n* lines from the top of the screen; –*n*

specifies the reserved line is displayed *n* lines from the bottom of the screen. For example, RESERVED –3 means the reserved line is three lines from the bottom of the screen.

Blue
Red
Pink
Green
Turquoise
Yellow
White
Default
are the choices for the color of the field. If a color is not specified, the default is DEFAULT (default base color).

BLInk
REVvideo (reverse video)
Underline
NONe
are the choices for the extended highlighting of the field. If an extended highlighting setting is not specified, the default is NONE (no extended highlighting).

PS0
PSA
PSB
PSC
PSD
PSE
PSF
are the choices for the programmed symbol set of the field. PS0 indicates using the default character set. If the programmed symbol set is not specified, PS0 is the default.

High
indicates the data in the reserved line is to be highlighted.

Nohigh
indicates the data in the reserved line is not to be highlighted (normal intensity).

text
specifies the information to be displayed in the reserved line.

Off
indicates a line reserved previously is to be returned for the editor's use.

Initial Setting

No reserved lines are defined.

Usage Notes

1. The QUERY RESERVED subcommand displays the line numbers of reserved lines.
2. If you are in a GUI environment, BLINK, PSS, and UNDERLINE are ignored.

Notes for Macro Writers

1. The EXTRACT/RESERVED/ subcommand returns line numbers of reserved lines. The EXTRACT/RESERVED */ subcommand returns information about reserved lines.

2. Reserved lines are associated with the file being edited when they are defined. They are displayed only when that file is being displayed.
3. SET RESERVED +*n* and SET RESERVED –*n* are considered to be two separate lines, even when they point to the same line on the screen.

   For example, assume you have a 43-line screen and issue the subcommands SET RESERVED 21 and SET RESERVED –4. Two different lines are reserved on this screen: line 21 and line 40. However, if the screen size changes to a 24-line screen, both reserved lines fall on the same line. In this case, the editor keeps both definitions but displays only the last one defined.

   When turning a reserved line off, you must specify the line the same way you defined it.
4. If you reduce the logical screen size (for example, by splitting the screen), only those reserved lines that fall within the smaller screen size are displayed. The definitions of lines that do not fit on the screen are kept, even though they are not displayed.
5. A SET RESERVED subcommand issued for the first line of the command line is ignored, unless SET CMDLINE OFF is in effect. This prevents you from accidentally losing the command line. A reserved line can be set on the second line of the command line if CMDLINE ON.
6. The default for reserved lines is protected. The SET CTLCHAR subcommand can be used to define the protection attribute. For more information on defining various attributes for fields within a reserved line, see “SET CTLCHAR” on page 258.
7. The SET RESERVED subcommand accepts the color, extended highlighting, and programmed symbol set operands regardless of whether the device could use those attributes. However, the action taken depends on the device. For example, HIGH and NOHIGH are ignored on a 3279 display (unless the color was DEFAULT), color is ignored on a 3278 display, and color, extended highlighting, and character set are ignored on a 3277 display. Also, EXTRACT returns all the settings, even those that may be ignored on any terminal.
8. On 3270 terminals equipped with the Programmed Symbol (PS) feature you can specify alternate character sets to be used on your terminal. The characters in these sets use different symbols, such as a different style or font, than the default character set.

   New character sets are loaded using the Graphic Data Display Manager (GDDM) licensed program (5748-XXH) or an application that loads programmed symbol sets. Load character sets before invoking XEDIT. Otherwise, if you load a new set, it is not recognized and the last character set loaded before invoking the editor is displayed; and, if you drop a programmed symbol set, the editor is not aware the set is no longer available and attempts to use that set. This may cause I/O errors when the display is written (resulting in SET TERMINAL TYPEWRITER). To avoid problems, do not load or delete program symbol sets when in the XEDIT environment.

   For more information on programmed symbol sets, see *IBM 3270 Information Display System Data Stream Programmer’s Reference*.
9. A SET RESERVED subcommand issued for the first line of the message line is ignored unless SET MSGLINE OVERLAY is in effect.
10. If you use SET RESERVED for screen row one, the reserved line is offset by one column; the row begins in column two instead of column one.
11. For more information on the effects of the SET RESERVED subcommand on wrapped lines, see page 368.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Responses

The information specified in the text operand, if any, appears on the specified reserved line.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
521E	**Invalid line number [RC=5]**
526E	**Option** *option* **valid in display mode only [RC=3]**
533E	**Line** *nn* **is not reserved [RC=4]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
3	Operand is valid only for display terminal
4	Line is not reserved
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SCALE

```
►►──┬─────┬──SCALe──┬─ON──┬──┬───────┬──────────────────────────────────────►◄
    └─SET─┘         └─OFF─┘  ├─M─────┤
                             ├─Mn────┤
                             ├─ M+n ─┤
                             ├─ M−n ─┤
                             ├─n─────┤
                             ├─ +n ──┤
                             └─ −n ──┘
```

Purpose

Use the SCALE option to control the display and position of the scale line on the screen. The scale line provides assistance while editing.

Operands

ON
displays the scale on the screen.

M specifies the line on which the scale line is displayed. The M stands for *middle of the screen* (rounded up for odd sized screens).

M*n*
M+*n*
M–*n*
You can combine M with a constant positive (+ is implicit) or negative integer to mean *n* lines below the middle of the screen (M+*n*), or *n* lines above the middle of the screen (M–*n*). For example, SCALE ON M means the *middle of the screen*, and SCALE ON M+3 means *three lines below the middle of the screen*.

n
+*n*
–*n* specifies the line on which the scale line is displayed. The *n* or +*n* (+ is implicit) specifies the scale line is displayed *n* lines from the top of the screen; –*n* specifies the scale line is displayed *n* lines from the bottom of the screen. For example, SCALE ON –3 displays the scale line three lines from the bottom of the screen.

OFF
removes the scale from the screen.

Initial Setting

```
SCALE ON M+1
```

Usage Notes

1. The scale looks like this:

```
<...+..|.1....+....2....+....3.>..+....4T...+....5....+....6....+....7...
.      .                        .        .
.      .                        .        .
.      .column pointer          .        .
.                               .        .truncation column
.left zone                      .right zone
```

2. If a line occupies more than one screen line and the scale is on an adjacent line, the scale is not displayed for that line. The scale is redisplayed when the line pointer moves to a file line that occupies only one screen line.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
521E	**Invalid line number [RC=5]**
526E	**Option** *option* **valid in display mode only [RC=3]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
3	Operand is valid only for display terminal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SCOPE

```
►►──┬─────┬──SCOPE──┬─Display─┬──────────────────────────────────────────────►◄
    └─SET─┘         └─All─────┘
```

Purpose

Use the SCOPE option to specify the set of lines on which the editor operates. The operand (ALL or DISPLAY) indicates if the collection includes all lines in the file or only the lines displayed. For an example of how to combine SET SCOPE effectively with the other selective line editing subcommands (SET DISPLAY, SET SHADOW, and SET SELECT), see "SET SELECT" on page 333.

Operands

Display
indicates the editor acts only on those lines currently in the display range as specified in SET DISPLAY. It performs as if the lines that are outside the display range are not part of the file.

All
indicates the editor acts on the entire file.

Initial Setting

```
SCOPE DISPLAY
```

Usage Notes

1. Target searches are performed only on lines within the current scope. If SCOPE DISPLAY has been set, the target search considers and looks at only displayed lines. This is true for all types of targets: absolute line numbers, a relative displacement from the current line, a line name, a simple string expression, or a complex string expression. Line movement through use of targets is done as if lines outside the scope had been removed. For example, NEXT or +1 may go from line 20 to line 40 if lines 21 to 39 are outside the display range.
2. If SCOPE DISPLAY is set and the current line's selection level is not within the SET DISPLAY *n1 n2* values, the editor forces the current line out of the display, causing the first line following it in the scope to be the new current line. This can occur if you issue:
   a. SET SELECT, changing the current line's selection level
   b. SET DISPLAY, changing lines that are displayed
   c. SET SCOPE DISPLAY and the previous setting was SCOPE ALL
3. If SCOPE ALL is set, the current line is always included in the display. Its selection level is not modified, but its display status is forced if it remains the current line.
4. While SCOPE ALL is set, the editor acts on the entire file, even that portion of the file that is not displayed. Therefore, the editor can change a line that is not displayed.
5. For examples of this and other selective line editing subcommands, examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID

is PREFIXX XEDIT), which excludes lines from the display. See also Appendix B, “Effects of Selective Line Editing Subcommands,” on page 455.

6. SORT, SET RANGE, and ALL work outside the scope.
7. If SCOPE DISPLAY is set and the last line of the file is a shadow line and the current line is the end of file line, entering input mode displays the last line of the shadow line group as the current line. In this case, a line that was outside the scope can be changed.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET SCREEN

```
                         (1)    ┌Horizontal┐
►►──┬─────┬──SCReen──┬──n──────┤          ├────────┬──────────────────────────►◄
    └─SET─┘          │          └Vertical──┘        │
                     │        ┌───────────┐         │
                     │        ▼       (2) │         │
                     ├─Size────sn─────────┴─────────┤
                     │        ┌───────────┐         │
                     │        ▼       (3) │         │
                     ├─Width───wn─────────┴─────────┤
                     │         ┌───────────────────┐│
                     │         ▼               (4) ││
                     └─Define───sln──swn──shn──svn─┴┘
```

Notes:

1 If specifying Horizontal, *n* is a minimum of 5 lines. If specifying Vertical, *n* is a minimum of 20 columns.

2 The minimum of *sn* is 5 lines.

3 The minimum of *wn* is 20 columns.

4 The maximum number of screens that can be created is 16, provided *sln* is not less than 5 and *swn* is not less than 20.

Purpose

Use the SCREEN option to divide the virtual screen into a specified number of logical screens so you can edit multiple files or multiple views of the same file. Each logical screen becomes, in effect, an independent terminal with its own file identification line, command line, and message line.

Operands

n specifies the number of logical screens the virtual screen is to be divided into. If only *n* is specified, HORIZONTAL is assumed.

Horizontal
specifies the logical screens are arranged horizontally, that is, one on top of the other. The *n* must be specified so all horizontal screens are at least five lines long. This is the default.

Vertical
specifies the logical screens are arranged vertically, that is, next to each other, from left to right. The *n* must be specified so all vertical screens are at least 20 columns wide.

Size *sn*
specifies the screens are created horizontally, where *sn* is the number of lines in each logical screen. Any number of screens can be created, provided each is at least five lines long (that is, *sn* cannot be less than five).

Width *wn*
specifies the screens are created vertically, where *wn* is the number of columns in each screen. Any number of screens can be created, provided each is at least 20 columns wide (that is, *wn* cannot be less than 20).

Define *sln swn shn svn*
indicates each screen is created according to the layout specified, where:

- *sln* is the number of lines in the logical screen.
- *swn* is the number of columns in the logical screen.
- *shn* is the line number of the upper left corner of the logical screen on the virtual screen.
- *svn* is the column number of the upper left corner of the logical screen on the virtual screen.

The maximum number of screens that can be created is 16, provided *sln* is not less than 5 and *swn* is not less than 20.

Initial Setting

`SCREEN SIZE` *sn*, where *sn* is the number of rows in the virtual screen.

Usage Notes

1. When you are editing multiple files, the files are arranged in a *ring* in virtual storage (see "XEDIT" on page 410 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 already displayed before the SET SCREEN subcommand was executed remain on the screen and keep their relative positions on the virtual screen.

   However, if the number of logical screens is decreased due to a SET SCREEN subcommand, files are still displayed. This begins with the file in the ring that issued the SET SCREEN subcommand when logical screens are available. Those files for which logical screens are no longer available are removed from the display.
2. When you define vertical screens (using SET SCREEN WIDTH or SET SCREEN DEFINE), the entire width of the virtual screen must be accounted for, that is, the logical screens must occupy the full virtual screen width.

   When defining vertical screens, remember to consider the width of the logical screens compared with the line length of the file(s) being edited. If the file line length exceeds the logical screen width, lines are displayed up to that width, that is, they do not *wrap,* and changing the SET VERIFY setting does not cause them to wrap.

   On vertical screens, messages are broken up into as many lines as needed to display the entire message. Changing the width of the logical screen while messages are pending may cause unpredictable results in the way the messages are displayed. If the number of message lines needed exceeds the number of lines specified in the SET MSGLINE subcommand, messages are passed to CMS to be displayed. When full-screen CMS is OFF, press CLEAR to redisplay the file. When full-screen CMS is ON, the message appears in the CMSOUT window. To see all the information in the virtual screen, you can use the CMS WINDOW FORWARD or WINDOW BACKWARD command. The screen is cleared automatically when you scroll to the bottom of the virtual screen. Alternately, you can clear the screen with the CMS WINDOW DROP command. If you have deleted the CMSOUT window, you do not see messages passed to CMS.

3. The SET SCREEN subcommand retains the CURLINE, SCALE, TABLINE, CMDLINE, MSGLINE, and RESERVED locations on the screens, if these settings fall within the new screen size. Otherwise, the default settings are used.

   When CMDLINE ON is in effect and vertical screens are defined, the command line is displayed on the bottom line of the screen because lines cannot extend on multiple screen lines (see Usage Note 2 on page 330). The status area is not displayed. The editor remembers this change and returns the CMDLINE to ON if the screen definition is subsequently changed to a nonvertical screen(s).
4. Entering input mode on one screen (issuing the INPUT subcommand without any operands) causes subcommands entered on other logical screens either to be ignored if the screen contains a view of the same file or to remain on the command line and not execute until input mode is exited if the screen contains a view of a different file.
5. For information on processing, see *z/VM: XEDIT User's Guide*, Chapter 6.
6. If more than one logical screen is defined before issuing a DISCONNECT from XEDIT, the screen setting is redefined to one screen when the session is reconnected on a different sized terminal. This would be the equivalent of issuing a `SET SCREEN 1` after reconnecting.
7. If the window being used is changed through the XEDIT window option while there are multiple logical screens, the editing session continues with only one logical screen (with the same number of rows and columns as the virtual screen).

Examples

This section shows examples of the SET SCREEN subcommand. For more information, see *z/VM: XEDIT User's Guide*.

Example 1

In this example, the screen is split horizontally into two logical screens.

```
set screen 2
```

Example 2

In this example, the screen is split vertically into two logical screens.

```
set screen 2 V
```

Example 3

In this example, the screen is split horizontally into two logical screens, the first of which is 14 lines long and the second of which is 10 lines long.

```
set screen size 14 10
```

Example 4

In this example, three logical vertical screens are created with the column widths of 25, 25, and 30, respectively.

```
set screen width 25 25 30
```

Example 5

In this example, entering

```
set scr def 16 40 1 1 16 40 1 41 8 80 17 1
```

results in the following screen layout on a 24 x 80 virtual screen.

```
    (1,1)                  (1,41)
      *************************************************
      *                       *                       *
      *                       *                       *
      *                       *                       *
      *        16 x 40        *        16 x 40        *
      *                       *                       *
      *                       *                       *
      *                       *                       *
(17,1)*************************************************
      *                                               *
      *                                               *
      *                    8 x 80                     *
      *                                               *
      *                                               *
      *************************************************
```

Responses

In horizontal screens, the status area of each logical screen contains the number of files being edited.

The screens are displayed as specified.

Messages and Return Codes

520E **Invalid operand:** *operand* **[RC=5]**
526E **Option** *option* **valid in display mode only [RC=3]**
534E **Too many logical screens defined [RC=4]**
536E **Logical screens exceed virtual screen size [RC=1]**
537E **Each logical screen must contain at least 5 lines and 20 columns [RC=4]**
543E **Invalid number:** *number* **[RC=5]**
545E **Missing operand(s) [RC=5]**
566E **Logical screen (***sln,swn,shn,svn***) is outside the virtual screen [RC=5]**
567E **Logical screens (***sln,swn,shn,svn***) and (***slm,swm,shm,svm***) overlap each other [RC=5]**
697E **The logical screens must cover the full virtual screen width [RC=5]**

where return codes are:

0	Normal
1	The total number of lines or columns for the multiple logical screens exceeds the virtual screen size
3	Operand is only valid for display terminal
4	Each logical screen must contain at least 5 lines and 20 columns, or too many logical screens defined
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SELECT

```
                                ┌─1──────┐
►►──┬─────┬──SELect──┬─n──┬──┼────────┼──────────────────────────────►◄
    └─SET─┘          ├─+n─┤  ├─target─┤
                     └─−n─┘  └─*──────┘
```

Purpose

Use the SELECT option to designate a selection level for specified lines. A selection level is a positive value assigned to a line in a file. You can logically group various lines in a file by assigning them the same selection level. This subcommand can be used effectively with the SET DISPLAY, SET SHADOW, and SET SCOPE subcommands.

Operands

n is the selection level for the lines specified. The *n* is a number; specified without a + or –, it assigns +*n* value as the selection level for the lines specified.

+*n*
–*n* adds (+) or subtracts (–) *n* from the current selection level(s) of the lines specified. For example, if you enter `SET SELECT 8 3` the selection level 8 is assigned to three lines in the file. To assign a level of 6 to the three lines to which you have just assigned a selection level of 8, enter `SET SELECT –2 3` or `SET SELECT 6 3`. Likewise, to assign a level of 10 to the three lines to which you have just assigned a selection level of 8, you could enter either `SET SELECT +2 3` or `SET SELECT 10 3`.

target
defines the number of lines to be assigned a selection level. The selection level is assigned from the current line up to, but not including, the specified target line. If *target* is not specified, only the current line's selection level is set.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

***** the selection level is assigned to the rest of the file.

Initial Setting

`SELECT 0` (for the entire file)

Usage Notes

1. Selective line editing can be used in a macro to control both the action of the editor and the screen display. It consists of four subcommands—SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW—that work together in the following way. You can use SET SELECT to assign a selection level, or value, to one or more lines in a file. Lines are logically grouped by assigning them the same selection level. SET DISPLAY can be used with SET SELECT to display those lines that have the same selection level. SET SCOPE defines the set of lines the editor can act on. SCOPE DISPLAY is the initial setting and, as such, restricts editor action to only those lines defined in SET DISPLAY. By default, SET SHADOW displays a notice indicating how many lines are not being

displayed in the physical position of the excluded lines in the file. If SET SHADOW is OFF, only those lines defined in SET DISPLAY appear on the screen, with no shadow lines to indicate where lines are not being displayed.

2. For examples of this and other selective line editing subcommands, you can examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display.
3. If you specify *–n*, which would cause the selection level of the line to be negative, the selection level is set to zero instead.
4. TOF and EOF are always set to the *n1* value of SET DISPLAY, even though you cannot assign a selection level to either of them.
5. For more information on selective line editing, see Appendix B, “Effects of Selective Line Editing Subcommands,” on page 455.

Examples

Figure 16 on page 335 demonstrates the selective line editing subcommands: SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE.

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

```
NICK      O for opera
          C for classical

COMPOSER  Name of Composer

NAME      Name of Composition

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

Elbert wanted to be able to list all the records in his collection by type. Using the macro below, Elbert located all the lines in the file that contained :nick.O and assigned a selection level of 1 to them. He also located all lines in the file that contained :nick.C, assigning a selection level of 2 to them.

```
/**** Xedit macro to set selection levels for Elbert's Records ...******/
'COMMAND TOP'                  /* Start at the Top of the file      */
'COMMAND SET WRAP OFF'         /* Do not wrap on Locate command!    */
'COMMAND SET MSGMODE OFF'      /* Suppress 'TARGET NOT FOUND' msg   */
'COMMAND SET CASE M I'         /* Ignore case on searches           */
'COMMAND LOCATE /:NICK.O/'     /* Look for first :nick.O            */
Do until rc ¬= 0               /* Loop until LOCATE fails (rc > 0)  */
  'COMMAND SET SELECT 1 1'     /* Set selection level 1             */
  'COMMAND UP 1'               /* Go back up one line               */
  'COMMAND LOCATE /:NICK.O/'   /* Now, find the next one...         */
  End                          /* End of loop                       */
'COMMAND TOP'                  /* Start at top again                */
'COMMAND LOCATE /:NICK.C/'     /* Now look for :nick.C              */
Do until rc ¬= 0               /* Loop until LOCATE  fails (rc>0)   */
  'COMMAND SET SELECT 2 1'     /* Set selection level               */
  'COMMAND UP 1'               /* Go back up one line               */
  'COMMAND LOCATE /:NICK.C/'   /* Find next occurrence              */
  End                          /* End of Loop                       */
'COMMAND TOP'                  /* Go back to the top                */
'COMMAND SET MSGMODE ON'       /* Now we want messages again        */
Exit                           /* Exit the macro                    */
```

Note: After the macro executes, you will be at the top of your file. To let you see more of the file, we have moved down in the file, making line 9 the current line. Also note, this macro is being executed against the base file *Elbert Names* shown on page 31.

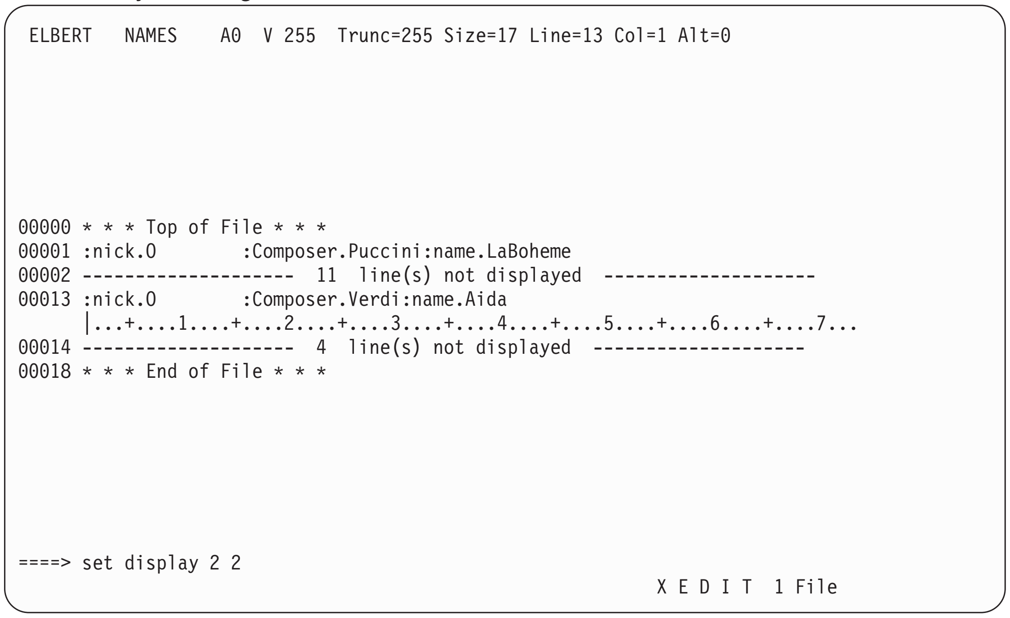
Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 1 of 9)

All nick.O or nick.C lines in the file have now been assigned a selection level. They appear as shadow lines, because the initial setting of SET DISPLAY is 0 0 and any line in the file not assigned a selection level of 1 or 2 has a selection level of 0. To look at all the nick.O compositions, Elbert entered SET DISPLAY 1 1, as shown on the command line.

```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=9 Col=1 Alt=0

00000 * * * Top of File * * *
00001 --------------------  1  line(s) not displayed  --------------------
00002               :addr.O1
00003
00004 --------------------  1  line(s) not displayed  --------------------
00005               :addr.C1
00006
00007 --------------------  1  line(s) not displayed  --------------------
00008               :addr.C4
00009
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00010 --------------------  1  line(s) not displayed  --------------------
00011               :addr.C3
00012
00013 --------------------  1  line(s) not displayed  --------------------
00014               :addr.O2
00015
00016 --------------------  1  line(s) not displayed  --------------------
00017               :addr.C2
00018 * * * End of File * * *
====> set display 1 1
                                                      X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 2 of 9)

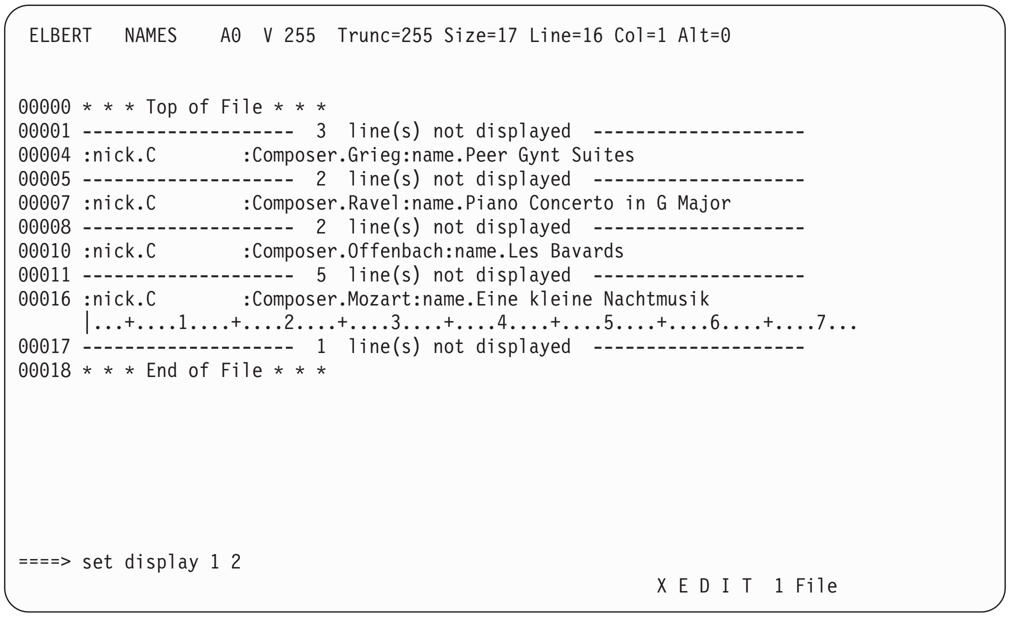
The resulting display listed only those lines with a selection level of 1 (those assigned to nick.O). Elbert then set the display to list all lines assigned a selection level of 2 by entering SET DISPLAY 2 2.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=13 Col=1 Alt=0









00000 * * * Top of File * * *
00001 :nick.O       :Composer.Puccini:name.LaBoheme
00002 --------------------  11  line(s) not displayed  --------------------
00013 :nick.O       :Composer.Verdi:name.Aida
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00014 --------------------  4  line(s) not displayed  --------------------
00018 * * * End of File * * *








====> set display 2 2
                                                          X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 3 of 9)

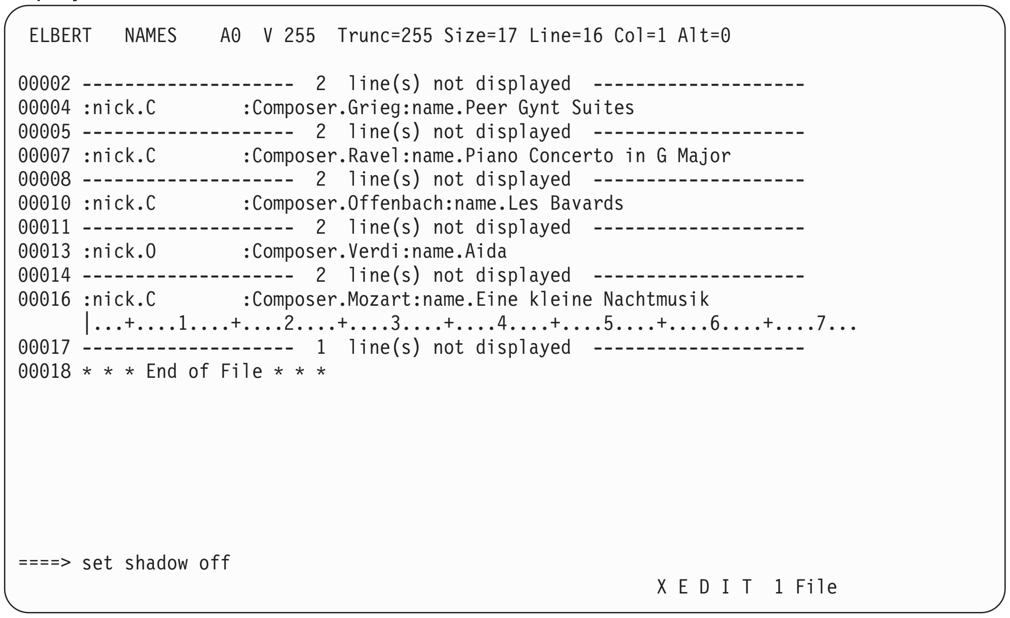
The display now listed only Elbert's classical records (nick.C). To list both types of records, Elbert entered SET DISPLAY 1 2 to display all lines in the file with either a selection level of 1 or 2.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0


00000 * * * Top of File * * *
00001 --------------------  3  line(s) not displayed  --------------------
00004 :nick.C       :Composer.Grieg:name.Peer Gynt Suites
00005 --------------------  2  line(s) not displayed  --------------------
00007 :nick.C       :Composer.Ravel:name.Piano Concerto in G Major
00008 --------------------  2  line(s) not displayed  --------------------
00010 :nick.C       :Composer.Offenbach:name.Les Bavards
00011 --------------------  5  line(s) not displayed  --------------------
00016 :nick.C       :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00017 --------------------  1  line(s) not displayed  --------------------
00018 * * * End of File * * *








====> set display 1 2
                                                          X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 4 of 9)

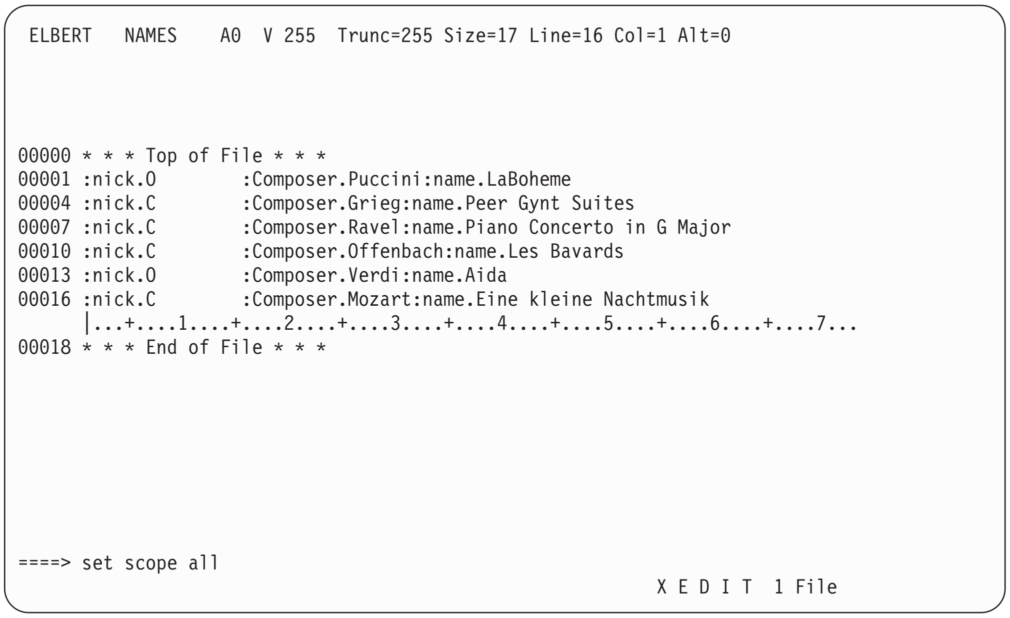
Elbert entered SET SHADOW OFF to eliminate shadow lines from the screen display.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0

00002 --------------------  2  line(s) not displayed  --------------------
00004 :nick.C        :Composer.Grieg:name.Peer Gynt Suites
00005 --------------------  2  line(s) not displayed  --------------------
00007 :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
00008 --------------------  2  line(s) not displayed  --------------------
00010 :nick.C        :Composer.Offenbach:name.Les Bavards
00011 --------------------  2  line(s) not displayed  --------------------
00013 :nick.O        :Composer.Verdi:name.Aida
00014 --------------------  2  line(s) not displayed  --------------------
00016 :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00017 --------------------  1  line(s) not displayed  --------------------
00018 * * * End of File * * *









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

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 5 of 9)

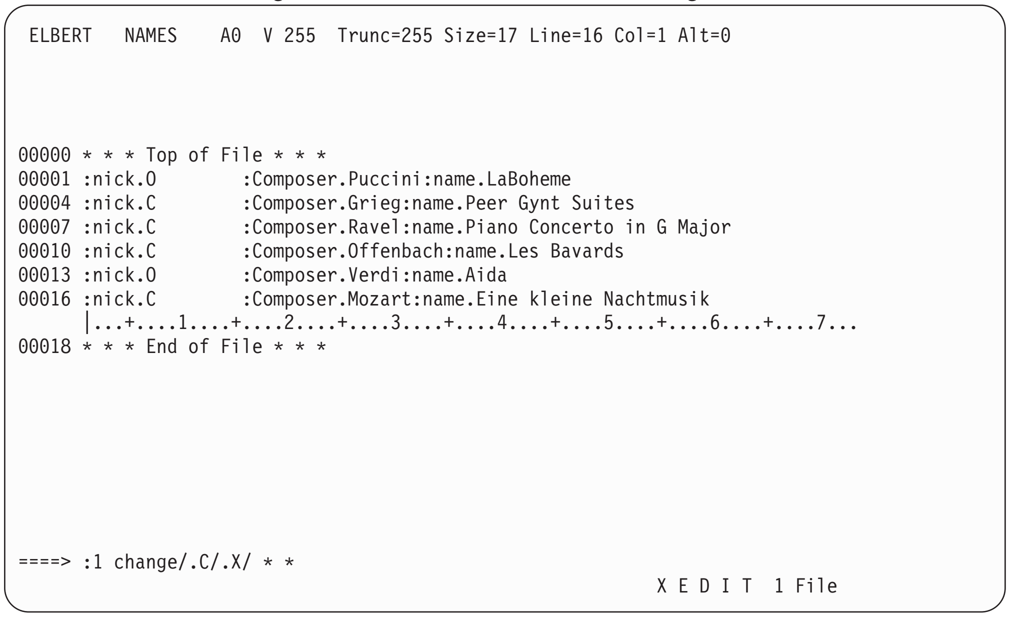
The result is a listing of all lines in the file that have been assigned selection levels. Note the absence of shadow lines.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0




00000 * * * Top of File * * *
00001 :nick.O        :Composer.Puccini:name.LaBoheme
00004 :nick.C        :Composer.Grieg:name.Peer Gynt Suites
00007 :nick.C        :Composer.Ravel:name.Piano Concerto in G Major
00010 :nick.C        :Composer.Offenbach:name.Les Bavards
00013 :nick.O        :Composer.Verdi:name.Aida
00016 :nick.C        :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00018 * * * End of File * * *









====> set scope all
                                                          X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 6 of 9)

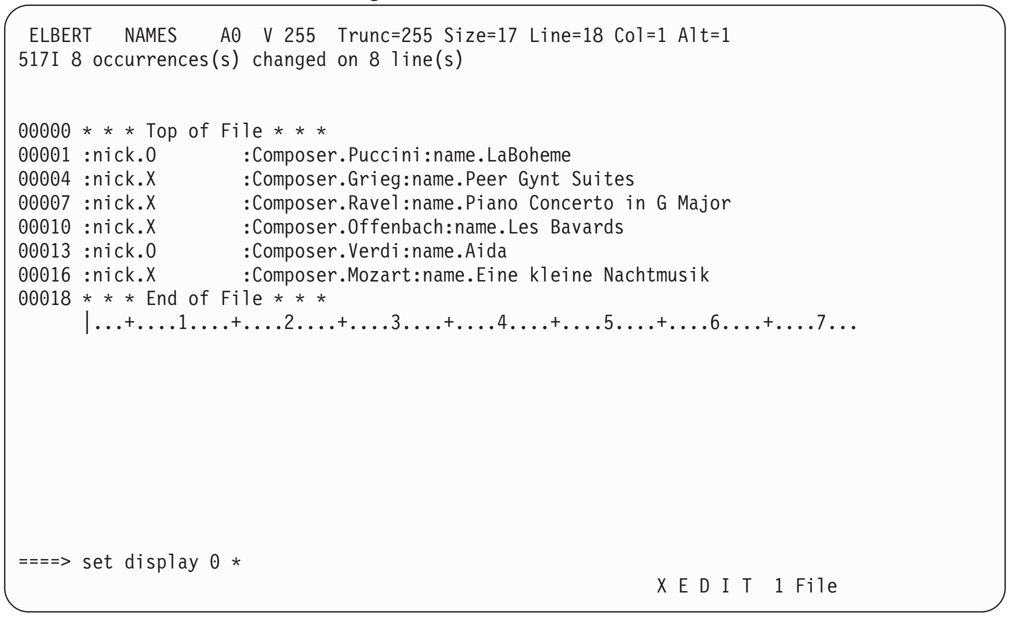
Elbert decided to add some country and western records to his collection. To avoid confusion between his classical records and his country records, he changed .C to .X. In the previous screen, he entered SET SCOPE ALL to expand editor control to include the entire file. This was necessary so both the lines listing composers' names and those listing record addresses would be changed.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=16 Col=1 Alt=0




00000 * * * Top of File * * *
00001 :nick.O       :Composer.Puccini:name.LaBoheme
00004 :nick.C       :Composer.Grieg:name.Peer Gynt Suites
00007 :nick.C       :Composer.Ravel:name.Piano Concerto in G Major
00010 :nick.C       :Composer.Offenbach:name.Les Bavards
00013 :nick.O       :Composer.Verdi:name.Aida
00016 :nick.C       :Composer.Mozart:name.Eine kleine Nachtmusik
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00018 * * * End of File * * *









====> :1 change/.C/.X/ * *
                                                            X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 7 of 9)

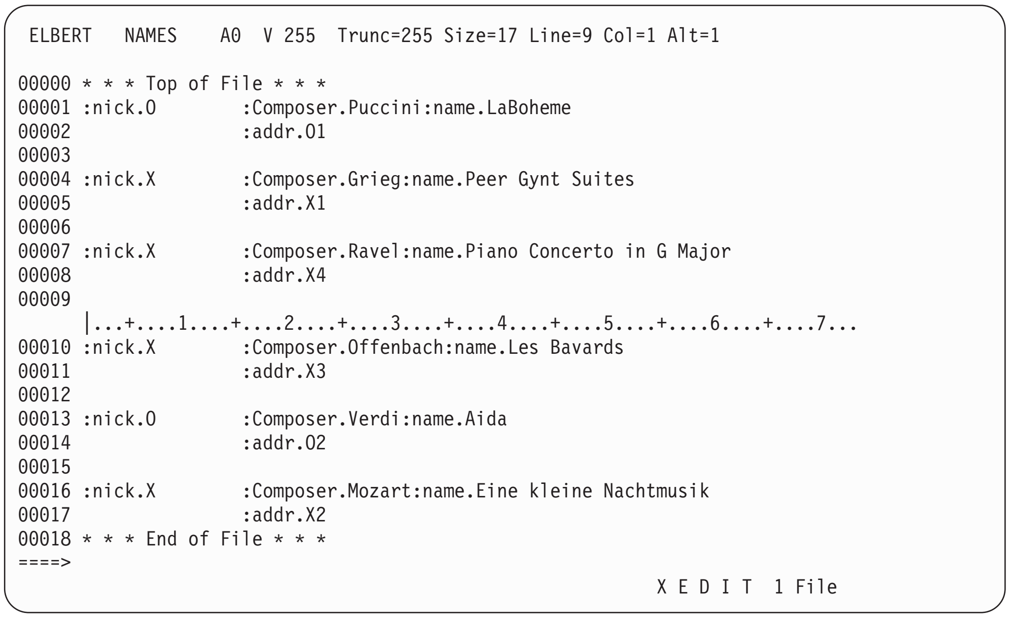
To display the entire file, Elbert entered SET DISPLAY 0 *. This displayed all selection levels of lines, starting with 0.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=18 Col=1 Alt=1
517I 8 occurrences(s) changed on 8 line(s)


00000 * * * Top of File * * *
00001 :nick.O       :Composer.Puccini:name.LaBoheme
00004 :nick.X       :Composer.Grieg:name.Peer Gynt Suites
00007 :nick.X       :Composer.Ravel:name.Piano Concerto in G Major
00010 :nick.X       :Composer.Offenbach:name.Les Bavards
00013 :nick.O       :Composer.Verdi:name.Aida
00016 :nick.X       :Composer.Mozart:name.Eine kleine Nachtmusik
00018 * * * End of File * * *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...










====> set display 0 *
                                                            X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 8 of 9)

After the changes, Elbert's file looks like the display below. Please note that we have changed the current line, without issuing the subcommand here, so you can see more of the file.



```
 ELBERT   NAMES    A0  V 255  Trunc=255 Size=17 Line=9 Col=1 Alt=1

00000 * * * Top of File * * *
00001 :nick.O        :Composer.Puccini:name.LaBoheme
00002                :addr.O1
00003
00004 :nick.X        :Composer.Grieg:name.Peer Gynt Suites
00005                :addr.X1
00006
00007 :nick.X        :Composer.Ravel:name.Piano Concerto in G Major
00008                :addr.X4
00009
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00010 :nick.X        :Composer.Offenbach:name.Les Bavards
00011                :addr.X3
00012
00013 :nick.O        :Composer.Verdi:name.Aida
00014                :addr.O2
00015
00016 :nick.X        :Composer.Mozart:name.Eine kleine Nachtmusik
00017                :addr.X2
00018 * * * End of File * * *
====>
                                                         X E D I T  1 File
```

Figure 16. SET SELECT, SET SHADOW, SET DISPLAY, and SET SCOPE (Part 9 of 9)

Responses

If you specify SET SELECT is to occur on multiple lines and it does occur, the current line pointer:

1. Is unchanged if SET STAY ON is in effect
2. Moves to the last line assigned a selection level, if SET STAY OFF is in effect (the default)

If SCOPE DISPLAY is set and the current line's selection level is not within the SET DISPLAY *n1 n2* values, the editor forces the current line out of the display, causing the first line following it in the scope to be the new current line. This can occur if you issue:

1. SET SELECT, changing the current line's selection level
2. SET DISPLAY, changing lines that are displayed
3. SET SCOPE DISPLAY and the previous setting was SCOPE ALL

Messages and Return Codes

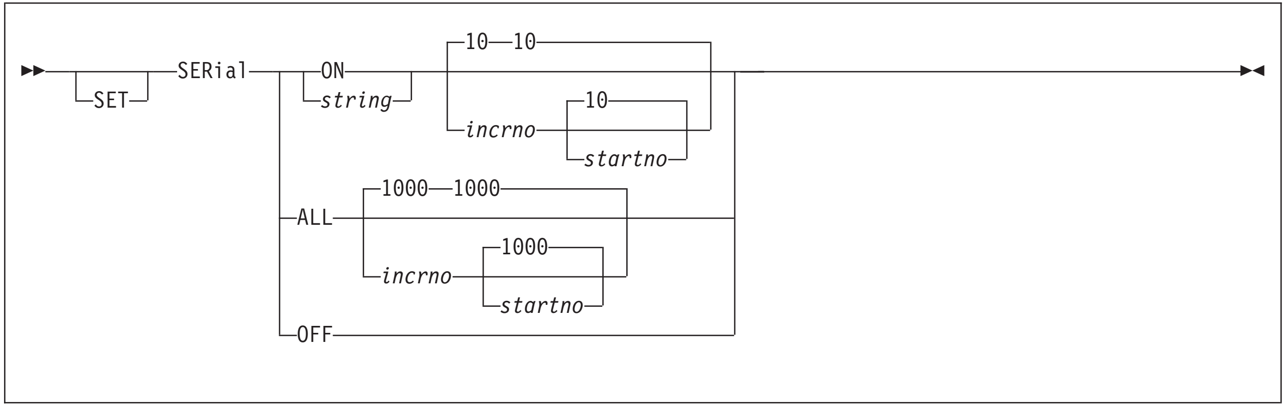
520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
546E	**Target not found [RC=2]**
585E	**No line(s) changed [RC=4]**

where return codes are:

0	Normal
1	TOF or EOF reached
2	Target not found
4	No change occurred
5	Invalid or missing operand(s)

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

SET SERIAL



Purpose

Use the SERIAL option to control file serialization.

Operands

ON
: adds a serial identification in the last eight columns of each file line. The serial identification consists of the first three letters of the file name plus five digits, where *startno* is the starting value and *incrno* is the increment.

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.

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.

OFF
: specifies the serial identification area is not to be updated the next time the file is written to disk or directory.

Initial Setting

Based on file type. See Appendix A, “File Type Defaults,” on page 453.

Usage Notes

1. Only files with a fixed record format can be serialized. Also, the room between TRUNC and LRECL must be at least eight characters to serialize the file (if not, message 560E is displayed).
2. The serialization takes place only when a FILE or SAVE subcommand is issued.
3. To remove the serial identification from a file whose logical record length is 80, you can use the following sequence of subcommands:

```
set trunc 80
set zone 1 80
set serial off
```

```
top
next
clocate :73
cdelete 8
repeat *
```

4. To renumber the line numbers of a VSBASIC or FREEFORT file, use the RENUM subcommand.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
545E Missing operand(s) [RC=5]
558E Wrong file format for serialization [RC=5]
560E Not enough space for serialization between TRUNC and LRECL

where return codes are:

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

SET SHADOW

```
►►──┬─────┬──SHADow──┬─ON──┬──────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

Use the SHADOW option to display a notice (called a shadow line) indicating how many lines are excluded from the display. The shadow line appears where the excluded line(s) would be in the file. For an example of how to combine SET SHADOW effectively with the other selective line editing subcommands (SET SELECT, SET SCOPE, and SET DISPLAY), see “SET SELECT” on page 333.

Operands

ON
displays a shadow line telling the number of lines omitted in the display by the SET DISPlay subcommand. Each shadow line shown is in the place of the omitted line or block of lines.

OFF
removes shadow lines, so there is no visual representation for lines not displayed.

Initial Setting

```
SHADOW ON
```

Usage Notes

1. For examples of this and other selective line editing subcommands, examine the following IBM-supplied macros: ALL (whose file ID is ALL XEDIT), which restricts editing to a particular set of lines, and the X prefix macro (whose file ID is PREFIXX XEDIT), which excludes lines from the display. See also Appendix B, “Effects of Selective Line Editing Subcommands,” on page 455.
2. If a prefix macro is entered on a shadow line, the line number of the first excluded line is passed to the macro as *pline* (the line number on which the prefix macro was entered). (See “SET PENDING” on page 304.)
3. With SET NUMBER ON, the sequence numbers let you determine which lines are excluded from the display. This is especially useful with SHADOW OFF. With SHADOW ON, the sequence number in a shadow line is the first line in a block of excluded lines.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET SIDCODE

```
                     (1)
                   ┌───────┐
►►──┬─────┬─SIDcode─┴───────┴──────────────────────────────────────────────►◄
    └─SET─┘          └─string─┘
```

Notes:

1 If *string* is not specified, SIDCODE is set to blanks.

Purpose

Use the SIDCODE option to insert a character string in every line of an update file. The string is inserted in the first eight columns of the last 17 columns of the file line (LRECL–16 to LRECL–9). For example, if you have a file with fixed, 80-character lines, the editor inserts the string in columns 64-71. If the string is fewer than eight characters, it is padded on the right with blanks. Any data in the eight columns is overlaid.

Operands

string
is the character string (one to eight characters) to be inserted in every line of an update file. If you omit *string*, SIDCODE is set to blanks. If *string* contains more than eight characters, it is truncated to eight characters.

Initial Setting

SIDCODE is set to blanks (or as specified in the XEDIT command or the LOAD subcommand).

Usage Notes

1. The string is inserted in every line of an update file when at least one change is made to the update file and it is filed (or saved).
2. Data will not be overlaid in lines with a character in column LRECL–8. Therefore, SIDCODE will not appear in the file or in the update file to indicate lines have been added or changed if those lines have a character in column LRECL–8.
3. See also the SIDCODE option of the XEDIT command.
4. When SIDCODE is set to all blanks (the default setting), columns LRECL–16 to LRECL–9 are unchanged.
5. Column LRECL–16 will not be overlaid if that position contains a % character.

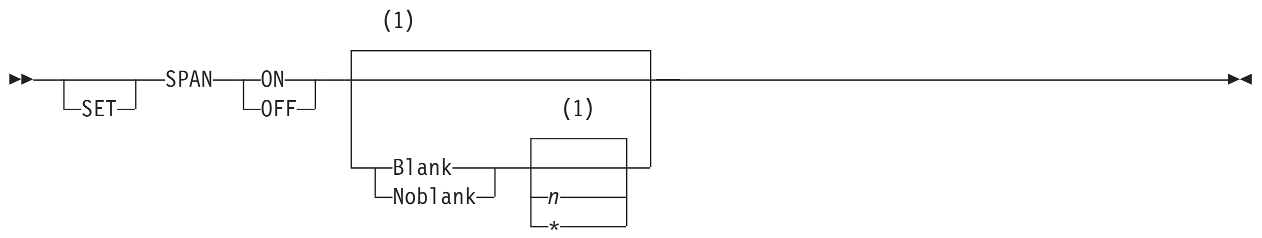
Messages and Return Codes

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

where return codes are:

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

SET SPAN



Notes:

1 The default is the previous setting.

Purpose

Use the SPAN option to specify if a character string that is the subject of a target search must be in one line to be found, or if it can span a specified number of lines.

Operands

ON
specifies lines are concatenated during a search for a character string. Trailing blanks are removed.

OFF
specifies a character string must be within one file line to match a target.

Blank
specifies 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 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.

***** searches the rest of the file.

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.

   Suppose a file contains on two consecutive lines:

left zone **right zone**
1 **72**

```
                    The house
       was near
```

If SET SPAN is ON, the two file lines, when concatenated for the search, 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, SET VARBLANK ON would permit the target `/house was/` to match the text.

On the other hand, a programmer editing an object deck produced by a compiler (file type TEXT, zones 1 72) or a PL/I source program (file type PLI, zones 2 72) should use SET SPAN ON NOBLANK, because no implicit blanks are assumed to separate lines.

For example:

left zone **right zone**
1 **72**

```
                                        X=' THE LIT
      TLE HOUSE';
```

The target `/THE LITTLE HOUSE/` would be found.

Examples

In the following example, SET SPAN tells the editor a string target can span two lines, separated from each other by a blank.

```
set span on blank 2
```

So, if you are looking for the string,

```
====> /house that Jack/
```

you would be able to locate it in this file even though the phrase was in two lines.

```
===== This is the house

===== that Jack built.
```

Messages and Return Codes

520E Invalid operand: ***operand*** **[RC=5]**
543E Invalid number: ***number*** **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET SPILL

```
►►──┬─────┬──SPILL──┬─ON───┬──────────────────────────────────────────────►◄
    └─SET─┘         ├─OFF──┤
                    └─WORD─┘
```

Purpose

Use the SPILL option to specify whether data is spilled onto new lines or lines are truncated after these subcommands: CHANGE, CINSERT, COVERLAY, CREPLACE, EXPAND, GET, INPUT, MERGE, OVERLAY, REPLACE, SHIFT, (and macros that use these subcommands internally, including CAPPEND, JOIN and PRFSHIFT (>, >>)).

Operands

ON
specifies characters pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character that would have gone beyond the truncation column.

OFF
specifies characters pushed beyond the truncation column are lost.

WORD
specifies characters pushed beyond the truncation column create one or more new lines, as necessary. However, a word is not split between lines; it is moved or *spilled* in its entirety. Thus, the first character of each new line is the first nonblank character following the last blank within the truncation setting (see "SET TRUNC" on page 364) on the affected line.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults," on page 453.

Usage Notes

1. SET SPILL affects only SHIFT issued with the RIGHT operand; any characters shifted past the truncation column spill onto new lines. SET SPILL has no effect on SHIFT LEFT; data shifted to the left past the zone1 column is lost.
2. For GET, the LRECL is used for the truncation column instead of the TRUNC setting.
3. For JOIN, SPILL OFF functions like SPILL ON. JOIN does not truncate data.
4. Recovered lines (see "RECOVER" on page 208) are not spilled.
5. New lines inserted because of the SPILL setting are always inserted starting at column 1, regardless of whether SET IMAGE is ON or OFF.
6. When SET VERIFY is ON and a subcommand changes a line that is spilled as a result, the last line inserted due to SET SPILL is displayed.

Examples

In the following example, the difference between SET SPILL WORD and SET SPILL ON is shown. When ON is entered, one or more new lines is created if necessary. The same happens when WORD is entered, however, the word is not split between the lines.

With SET SPILL WORD:
```
===== Now is the time for all good men to come to the aid of their party.

====> c/good/good, courageous, stout-hearted/
```

results in:
```
===== Now is the time for all good, courageous, stout-hearted men to come to
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== the aid of their party.
```

With SET SPILL ON, the same CHANGE subcommand results in:
```
===== Now is the time for all good, courageous, stout-hearted men to come to t
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== he aid of their party.
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET STAY

```
►►──┬─────┬──STAY──┬─ON──┬──────────────────────────────────────────────────────►◄
    └─SET─┘        └─OFF─┘
```

Purpose

Use the STAY option to specify whether the line pointer is to move when a string that is the object of a LOCATE target search, FIND, FINDUP, NFIND, or NFINDUP is not found. Also, use the STAY option to specify whether the line pointer is to move when a string that is the object of the target search with CHANGE, COUNT, COMPRESS, EXPAND, LOWERCAS, SET SELECT, SHIFT, or UPPERCAS is found.

Operands

ON
specifies the line pointer is *not to move*.

OFF
specifies the line pointer moves.

Initial Setting

```
STAY  OFF
```

Usage Notes

1. SET STAY applies to LOCATE target searches and FIND family searches issued to the end of file (or end of range). With SET STAY OFF, the null End of File (or End of Range) line becomes the new current line if a search is unsuccessful. The Top of File (or Top of Range) line becomes the new current line if the search was in a backward direction. With SET STAY ON, the current line remains the same.
2. SET STAY also applies to the following subcommands: CHANGE, COUNT, COMPRESS, EXPAND, LOWERCAS, SET SELECT, SHIFT, and UPPERCAS. With SET STAY OFF, the last line examined or acted upon becomes the new current line; with SET STAY ON, the line pointer does not move when the subcommand is executed.

Messages and Return Codes

520E **Invalid operand:** *operand* **[RC=5]**
545E **Missing operand(s) [RC=5]**

where return codes are:

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

SET STREAM

```
►►──┬─────┬──STReam──┬─ON──┬──────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

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.

Operands

ON
specifies the editor begins searching at the character following the column pointer and continues to the end of file (or 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 file (or range)).

OFF
specifies the editor searches only the current line (within the limits defined in the SET ZONE subcommand).

Initial Setting

```
STREAM  ON
```

Messages and Return Codes

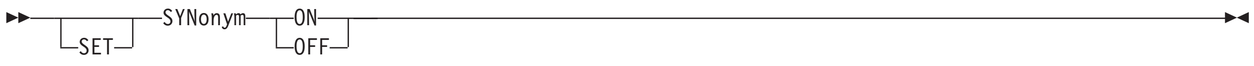
520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

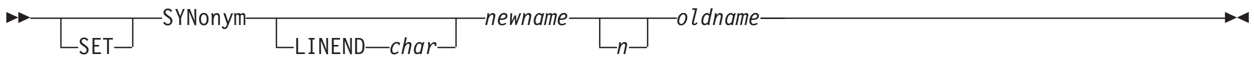
0	Normal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET SYNONYM

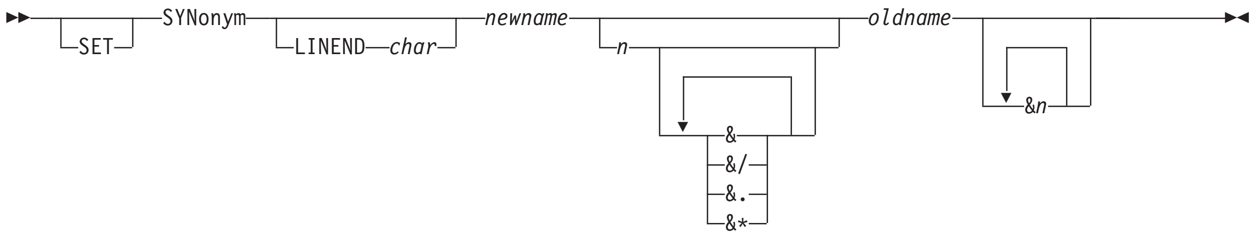
Format 1



Format 2



Format 3



Purpose

The SET SYNONYM subcommand has three formats.

Use Format 1 to specify whether the editor is to look for synonyms.

Use Format 2 to assign a synonym to any existing subcommand or macro (except prefix subcommands or prefix macros) and, optionally, to define an abbreviation for the synonym. (You must use the SET PREFIX subcommand to define a synonym for a prefix subcommand or macro.)

Use Format 3 to automatically rearrange operands in the order XEDIT expects. Do this when a synonym represents a subcommand whose operands are entered in a different order.

Operands

ON
specifies the editor is to look for synonyms.

OFF
specifies the editor is not to look for synonyms.

LINEND *char*
specifies a character that is interpreted as a line-end character regardless of the current SET LINEND setting at the time the synonym is used.

newname
is the synonym to be assigned to the subcommand or macro. The synonym can be an alphabetic string from 1 - 8 characters, or it can be a single alphabetic, numeric, or special character.

n
is the minimum number of characters you can enter for the synonym to be accepted, that is, its minimum abbreviation. If *n* is not specified, the full synonym name (*newname*) is required.

&
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 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 can be a compound name (for example, QUERY PF) or a subcommand name followed by its arguments.

&*n*
specifies the relative order which the new operands are to be inserted in the XEDIT subcommand, even though they are entered in a different order with the synonym. An &1 would represent the first operand in the synonym operand list, an &2 would 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:

```
SET SYNONYM ALTER      2 ALTER
SET SYNONYM CAPPEND    2 CAPPEND
SET SYNONYM FILE       4 COMMAND PFILE
SET SYNONYM SSAVE      2 COMMAND SAVE
SET SYNONYM FFILE      2 COMMAND FILE
SET SYNONYM HELP       1 HELP
SET SYNONYM HEXTYPE    4 HEXTYPE
SET SYNONYM JOIN       1 JOIN
SET SYNONYM MODIFY     3 MODIFY
SET SYNONYM QUIT       4 COMMAND PQUIT
SET SYNONYM QQUIT      2 COMMAND QUIT
SET SYNONYM SAVE       4 COMMAND PSAVE
SET SYNONYM SPLIT      2 SPLIT
SET SYNONYM STATUS     4 STATUS
```

Usage Notes

1. The *newname* operand can be the name of an existing XEDIT subcommand. In this case, the SYNONYM subcommand defines a new meaning for that subcommand name. The original meaning can be obtained by using:

   `command` *oldname* ...

   or

   ```
   set  synonym  off
   ```
   oldname...
2. The *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. Do not define a synonym for a name already defined as a synonym. For example:

```
synonym erase delete
synonym remove erase
```

   If you enter REMOVE, the editor looks for a subcommand called ERASE, not for a subcommand called DELETE.

```
synonym add delete
synonym linend $ SXMS locate/A/$ADD
```

   If you enter `SXMS`, a line is added after the line containing string A. The DELETE does not occur.

Examples

This section shows examples of the SET SYNONYM subcommand.

Example 1

In the following example, the Format 2 of SYNONYM is shown.

1. `syn  down  1  up`
2. `syn  qpf  query  pf`
3. `syn  linend | QK 2 query pf|query pa|query enter`

   Entering `QK` displays the PF, PA, and ENTER key settings.

Example 2

In the following example, the Format 3 of SYNONYM is shown.

1. `syn  putfile  3  &  &  &  &  put  &4  &1 &2  &3`

   If you enter the command as follows:

   `put` *fn ft fm n*

   The editor rearranges it as follows:

   `put` *n fn ft fm*

2. `syn  alter  2  &.  &/  &*  change  /X'&1'/X'&2'/&3`

   This example translates the subcommand:

   `alter  /7B/15/  *  *`

   to the following:

   `change  /X'7B'/X'15'/  *  *`

Messages and Return Codes

497E Minimum abbreviation is between SO and SI [RC=5]
520E Invalid operand: *operand*
544E Invalid hex data: *xxxxxxxx* **[RC=5]**
545E Missing operand(s) [RC=5]
547E Synonym definition incomplete [RC=5]
548E Invalid synonym operand: *operand* **[RC=5]**
549E Synonym abbreviation too large [RC=5]
550E Too many operands in synonym definition [RC=5]

where return codes are:

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

SET TABLINE

```
                                         (1)
                                      ┌───────┐
►►──┬─────┬──TABLine──┬─ON──┬─────────┼───────┼──────────────────────────────────────────►◄
    └─SET─┘           └─OFF─┘         ├─M─────┤
                                      ├─Mn────┤
                                      ├─ M+n ─┤
                                      ├─ M−n ─┤
                                      ├─n─────┤
                                      ├─ +n ──┤
                                      └─ −n ──┘
```

Notes:

1 The default is the previous setting.

Purpose

Use the TABLINE option to display, on a specified line, a T in every tab column according to the current tab settings (see "SET TABS" on page 356).

Operands

ON
displays the tab line.

OFF
removes the tab line from the screen.

M specifies the line on which the tab line is displayed. The M stands for *middle of the screen* (rounded up for odd-sized screens).

M*n*
M+*n*
M–*n*
You can combine M with a constant positive (+ is implicit) or negative integer to mean *n* lines below the middle of the screen (M+*n*), or *n* lines above the middle of the screen (M–*n*).

n
+*n*
–*n* specifies the line where the tab line is displayed. The *n* or +*n* (+ is implicit) specifies the tab line is displayed *n* lines from the top of the screen; –*n* specifies the tab line is displayed *n* lines from the bottom of the screen.

Initial Setting

`TABLINE OFF -3` (three lines from the bottom of the screen).

Usage Notes

1. TABLINE can be set on the same line as the scale line (see "SET SCALE" on page 325).
2. If a line occupies more than one screen line and the tab line is set on an adjacent line, the tab line is not displayed. The tab line is redisplayed when the line pointer moves to a file line that occupies only one screen line.

Examples

This section shows examples of the SET TABLINE subcommand. For more information, see *z/VM: XEDIT User's Guide*.

Example 1

In the following example, the tab line is displayed in the middle of the screen.

```
tabline on m
```

Example 2

In the following example, the tab line is displayed three lines below the middle of the screen.

```
tabline on m+3
```

Example 3

In the following example, the tab line is displayed three lines from the bottom of the screen.

```
tabline on -3
```

Responses

The specified line contains a T in every tab column. For example:

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

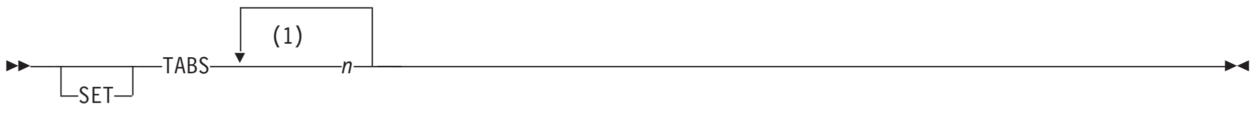
Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
521E Invalid line number [RC=5]
526E Option *option* **valid in display mode only [RC=3]**
545E Missing operand(s) [RC=5]

where return codes are:

0	Normal
3	Operand is valid only for display terminal
5	Invalid or missing operand(s) or number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET TABS



Notes:

1 You can specify *n* up to 28 times.

Purpose

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

Operands

n define the column numbers for logical tab settings. You can specify up to 28 numbers, separated from each other with at least one blank.

Initial Setting

Based on file type. See Appendix A, “File Type Defaults,” on page 453.

Usage Notes

1. SET IMAGE ON 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. SET FILLER defines the *space* character; the 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 tab settings. (See “COMPRESS” on page 62 and “EXPAND” on page 86.)
4. On a display terminal, the SET TABS subcommand controls the *logical* tab settings and the *physical* tab settings. You can set up a PF key to function like a tab key on a typewriter (see SET PF*n* TABKEY); each time you press the PF key, the cursor moves to the next column defined in SET TABS.
5. Default tab settings differ according to file type. QUERY TABS displays them.
6. To define a tabulation character, issue the CMS command SET INPUT. For example, the following command defines a > as the tabulation character:

   ```
   cms set input > 05
   ```

Examples

For more information, see *z/VM: XEDIT User’s Guide*.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]
575E Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} columns defined [RC=5]

where return codes are:

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

SET TERMINAL

```
►►──┬─────┬──TERMinal──┬─Typewriter─┬────────────────────────────────────────────────►◄
    └─SET─┘            └─Display────┘
```

Purpose

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.)

Operands

Typewriter
specifies the display terminal is to be used in line mode.

Display
specifies the terminal is to be used in full-screen mode.

Initial Setting

The initial setting is based on whether your terminal was disconnected or connected during your XEDIT session. If disconnected, the initial setting is TERMINAL TYPEWRITER; if connected, the initial setting is TERMINAL DISPLAY. However, if you are in a GUI environment, the initial setting is TERMINAL DISPLAY regardless of your connection.

Usage Notes

1. With a remote display terminal, full-screen performance depends on the line transmission rate; if it is too slow, you can specify line mode (TYPEWRITER).
2. In case of a severe transmission error while in full-screen mode (DISPLAY), the editor automatically switches to line mode (TYPEWRITER).
3. If you are editing a file in full-screen mode and you issue a DISCONN (disconnect) command, you must issue BEGIN after you reconnect, and then press ENTER to get the file image back on the screen.
4. If you are editing a file in line mode (SET TERMINAL TYPEWRITER), XEDIT does not recognize which key caused an attention interrupt. The definition that is executed is the CP definition of that key. To QUERY keys in line mode, use the CP QUERY PF*nn* command.
5. If you are in a GUI environment, TERMINAL is DISPLAY regardless of whether the emulator is connected or NOSCREEN has been specified. Use the REXX DIAG 24 function to determine if the emulator is connected.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
526E	**Option** *option* **valid in display mode only [RC=3]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

0	Normal
3	Operand is valid only for display terminal
5	Invalid or missing operand(s)

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

SET TEXT

```
►►──┬─────┬──TEXT──┬─ON──┬──────────────────────────────────────────────────────►◄
    └─SET─┘        └─OFF─┘
```

Purpose

Use the TEXT option to inform the editor and CMS if TEXT characters are to be used.

Operands

ON
specifies TEXT characters are to be used. You must issue a SET TEXT ON before using these keys so proper character code conversion takes place.

OFF
specifies no code conversion is to be performed for TEXT keys.

Initial Setting

Defined by the CMS setting.

Usage Notes

1. If a terminal has the TEXT feature, special TEXT keys are available. Before using these keys, you must inform the editor so proper character code conversion takes place. There are two ways to inform the editor. Issue:
   a. CMS SET TEXT ON
   b. The editor SET TEXT ON subcommand
2. Because the conversion is costly, it is recommended you issue the XEDIT subcommand SET TEXT OFF when you stop using the special keys.
3. Issuing SET TEXT ON while APL is ON causes APL to be set to OFF. Similarly, issuing SET APL ON while TEXT is ON causes TEXT to be set to OFF.
4. Changing the TEXT setting for XEDIT also changes the TEXT setting for CMS.
5. If you are in a GUI environment, SET TEXT is ignored.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET TOFEOF

```
►►──┬─────┬──TOFEOF──┬─ON──┬──────────────────────────────────────────────►◄
    └─SET─┘          └─OFF─┘
```

Purpose

Use the TOFEOF option to control the display of the following notices:

Top of File
End of File
Top of Range
End of Range

Operands

ON
specifies the notices listed above are displayed.

OFF
specifies the notices listed above are not displayed.

Initial Setting

```
TOFEOF ON
```

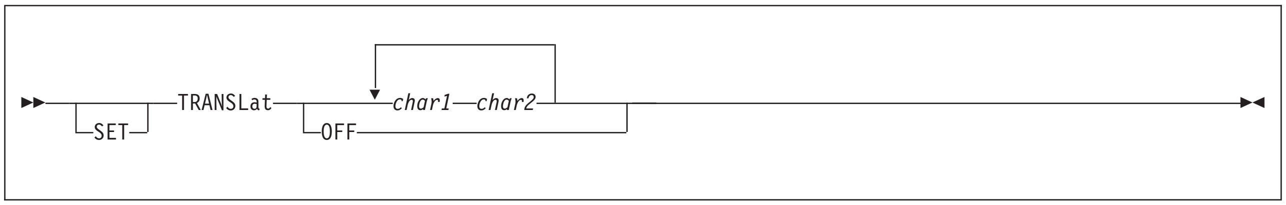
Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET TRANSLAT



Purpose

Use the TRANSLAT option to control uppercase translation of specified characters. This option is designed for use on terminals whose keyboards support characters other than English. By default, only English alphabetic characters are translated to uppercase.

Operands

char1 char2
: define a lowercase/uppercase pair of characters. The *char1* is the lowercase character, and *char2* is the uppercase character. You can specify either as a single EBCDIC character or a two-digit hexadecimal character.

OFF
: indicates characters are not to be translated to uppercase.

Initial Setting

Only English characters are translated to uppercase.

Usage Notes

1. For data lines, translation occurs only if SET CASE UPPER is in effect or if a line(s) is changed to uppercase (with the UPPERCAS subcommand, for example). An existing data line is not translated to uppercase unless a change is made to the line, in which case the entire line is translated.
2. Commands are translated if SET CASE UPPER is in effect.
3. Translation specified with SET TRANSLAT occurs only for uppercase. For lowercase, the system-supplied translate table is used.

Examples

In the following example, assume the terminal keyboard has the French character set and SET CASE UPPER:

```
set translat 51 E
```

(X'51' is “e” acute.)

User enters:

```
      / /     /          /
liberte egalite fraternite
```

which results in:

```
LIBERTE EGALITE FRATERNITE
```

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
545E Missing operand(s) [RC=5]

where return codes are:

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

SET TRUNC

```
►►──┬─────┬──TRunc──┬─n─┬──────────────────────────────────────────────────►◄
    └─SET─┘         └─*─┘
```

Purpose

Use the TRUNC option to define the truncation column, which is the last column in a line you can enter or modify data.

Operands

n	specifies the column at which truncation (or *spilling* — see “SET SPILL” on page 347) occurs.
*****	the truncation column is set to the record length for the file type.

Initial Setting

- For SFS and minidisk fils, the initial settings are based on the file type. See Appendix A, “File Type Defaults,” on page 453.
- For BFS files, the initial setting is the same as the WIDTH.

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 LRECL–8.

Messages and Return Codes

009E	**Column** *nn* **exceeds record length (***nn***) [RC=5]**
520E	**Invalid operand:** *operand* **[RC=5]**
543E	**Invalid number:** *number* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET VARBLANK

```
►►──┬─────┬──VARblank──┬─ON──┬─────────────────────────────────────────────────────────────────►◄
    └─SET─┘            └─OFF─┘
```

Purpose

Use the VARBLANK option to control whether the number of blank characters between two words is significant in a target search.

Operands

ON
specifies the number of blanks between two words can be variable and does not matter in searching for a target.

For example:

```
/the  house/
```

matches in the text:

```
the      house
```

and also matches:

```
the            house
```

OFF
specifies 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.

Examples

In the following example, SET VARBLANK ON is issued so the number of blanks between two words can vary; the number of intervening blanks specified in a string target does not have to be equal to the number in the file.

```
set varblank on

====> /cats dogs/
```

would locate either of the following lines in a file:

```
===== cats     dogs

===== cats dogs
```

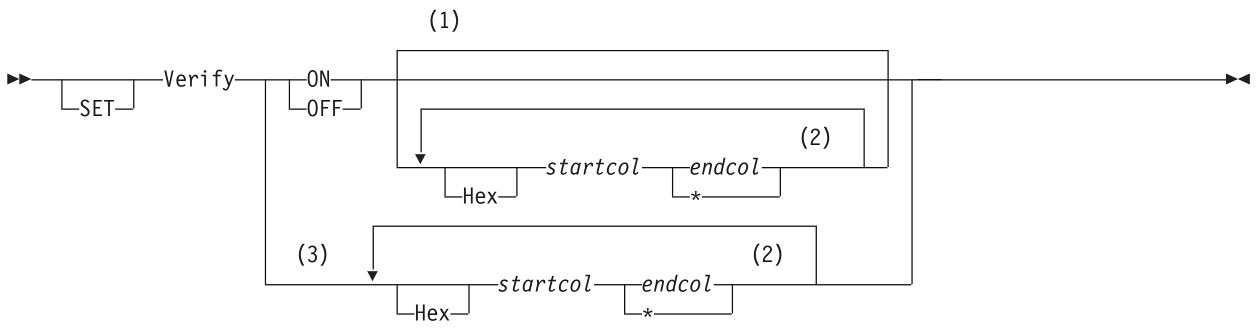
Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET VERIFY



Notes:

1 The default is the previous setting.

2 You can specify up to 28 pairs of columns.

3 The previous ON or OFF setting is retained.

Purpose

Use the VERIFY option to:

- Control whether lines that subcommands change are to be displayed (in the message area, for a display terminal). If the current line pointer changes, the new current line is displayed.
- Define the columns to be displayed when a file appears on the screen. Optionally, data may be displayed in hexadecimal notation.

Operands

ON
specifies all lines changed by subcommands are to be displayed. (This is the initial setting for a typewriter terminal.)

OFF
specifies lines changed by subcommands are not to be displayed. (This is the initial setting for a display terminal.)

Hex
displays the data in hexadecimal notation.

startcol endcol
are a pair of column numbers that define an area to be displayed.

***** indicates LRECL will be used for *endcol*.

Initial Setting

Based on file type. See Appendix A, "File Type Defaults," on page 453.

Usage Notes

1. You can specify up to 28 pairs of columns. For example:

```
v 1 20 40 50
```

displays columns 1 through 20 and 40 through 50.

2. You can display an area in both EBCDIC and hexadecimal. For example:

   ```
   v 1 20 H 1 20
   ```

   displays columns 1 through 20 in both EBCDIC and hexadecimal.
3. You can change the data in either the EBCDIC or the hexadecimal display. To change the hexadecimal display, enter changes in hexadecimal. If you type an invalid hexadecimal code, the following message is displayed on the first line of the screen (the file identification line):

   ```
   Invalid hex-data on screen:
   ```
4. The SET IMAGE setting affects the way hexadecimal data is treated. For example, if IMAGE is ON and you enter a X'05', it expands into X'40's (or to whatever the FILLER character is in hexadecimal) to the next tab position.
5. In multiple views of the same column, if changes are made on the screen the right-most change is the effective one.
6. The columns specified in a SET VERIFY subcommand override any RIGHT or LEFT subcommand in effect.
7. The editor displays a file line on as many screen lines as necessary. You can turn off this *automatic line wrapping* feature by issuing the appropriate SET VERIFY subcommand.

   For example, the default VERIFY setting for a SCRIPT file type is 1-132. To display only columns 1-72 of each line, thereby preventing lines longer than 72 characters from wrapping around to the next screen line, you could issue SET VERIFY 1 72. (You could then use a RIGHT subcommand to view the columns of data extending past column 72.)

   The message lines or reserved lines may effect the number of wrapped lines that are displayed.

   Automatic line wrapping is not available on vertical screens.

   In a GUI environment, to improve performance choose a verify setting that prevents automatic line wrapping.
8. For typewriter terminals, the maximum verify width is 132.
9. Adding characters or blanks using the terminal's Insert key will result in lost data. Use XEDIT subcommands, such as CINSERT to avoid lost information.

   For example, if you specify `v 1 20` and the result is:

   ```
          |....+....1....+....2....+....3....+....4....+

   00001   This is an example o
   ```

   Inserting the characters '1. ' using the terminal's Insert key and pressing the Enter key would result in:

   ```
          |....+....1....+....2....+....3....+....4....+

   00001   1. This is an exampl
   ```

   The characters 'e o' will no longer be contained in the line and will not be displayed even if the verify columns are expanded to display additional columns. For more information on the CINSERT subcommand, see "CINSERT" on page 50.

Messages and Return Codes

009E Column *nn* exceeds record length (*nn*) [RC=5]

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

545E Missing operand(s) [RC=5]

575E **Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} column(s) defined [RC=5]**
576E **{Total verify width exceeds screen size (*nn*)|Total offset exceeds LRECL (*nn*)} [RC=5]**
581E **Subcommand is not valid in extended mode [RC=3]**

where return codes are:

0	Normal
3	Subcommand is not valid in extended mode
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SET WRAP

```
►►──┬─────┬──WRap──┬─ON──┬──────────────────────────────────────────────────────►◄
    └─SET─┘        └─OFF─┘
```

Purpose

Use the WRAP option to control whether 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.

Operands

ON
specifies the editor is to “wrap around” the file if the end (or top) of file (or range) is reached.

OFF
specifies the editor is to stop searching at the end (or top) of file (or range).

Initial Setting

```
WRAP  OFF
```

Usage Notes

1. When a LOCATE is issued with SET WRAP ON, the search continues up to the line preceding the current line (or following the current line, depending on the search direction).

   When a CLOCATE is issued with SET WRAP ON, the search continues up to the character preceding the column pointer (or following the column pointer).

   When a FIND, FINDUP, NFIND, or NFINDUP is issued with SET WRAP ON, the search will continue through the entire file and stop when the line where it started is reached again.

Messages and Return Codes

520E	**Invalid operand:** *operand* **[RC=5]**
545E	**Missing operand(s) [RC=5]**

where return codes are:

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

SET ZONE

```
►►──┬─────┬──Zone──zone1──┬─zone2─┬──────────────────────────────────────────────►◄
    └─SET─┘               └─*─────┘
```

Purpose

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.

Operands

zone1
is the starting (left) column number of the zone.

zone2
is the ending (right) column number of the zone.

* if specified, *zone2* is the truncation column. (*Zone2* cannot be larger than the truncation column.)

Initial Setting

The ZONE is set from the first tab column up to the truncation column.

Usage Notes

1. The zone definition limits column pointer movement: 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 SET SPAN ON NOBLANK is in effect, column *zone2* of the current line is immediately followed by column *zone1* of the next line, with no intervening blank.

   If SET SPAN ON BLANK is in effect, 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.

Responses

When a SET ZONE subcommand is issued, the column pointer movement:

- Remains unchanged, if the column pointer already lies between the two new zones
- Moves to the column defined by *zone1*–1 (TOL), if the current setting of the column pointer is less than *zone1*
- Moves to the column defined by *zone2*+1 (EOL), if the current setting of the column pointer is greater than *zone2*

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
545E Missing operand(s) [RC=5]

575E Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} columns defined [RC=5]

where return codes are:

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

SET =

```
►►──SET──=──string─────────────────────────────────────────────────────────►◄
```

Purpose

Use the = option to insert the specified string into the equal (=) buffer. (See "= (Equal Sign)" on page 416.)

Operands

string
is any XEDIT subcommand or macro, except for the = subcommand (or any CP or CMS command, if SET IMPCMSCP is in effect).

Usage Notes

1. The subcommand SET is required with this option (to avoid conflict with the = subcommand).

Messages and Return Codes

545E Missing operand(s) [RC=5]

where return codes are:

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

SHIFT

```
                    ┌─1──1──────────────────┐
►►──SHift──┬─Left──┬┴───────────────────────┴─────────────────────────────────────────►◄
           └─Right─┘└───cols───┬─1──────────┤
                               ├─target─┤
                               └─*──────┘
```

Purpose

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

Operands

Left
shifts data to the left. Data shifted to the left past the zone1 column is lost. The line is padded with blanks to the right, up through the truncation column.

Right
shifts data to the right. Shifted data that extends past the truncation column may be lost (See Usage Note 2). The line is padded to the left with blanks.

cols
is the number of columns the data is to be shifted. The default is 1.

target
defines the number of lines to be shifted, starting with the current line, up to but not including the target line. If *target* is not specified, only the current line is shifted.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

***** the rest of the file is shifted.

Usage Notes

1. The SHIFT subcommand should not be confused with LEFT, RIGHT, and SET VERIFY, which move the screen over the data, causing the data to *appear* to move in the opposite direction, and do not cause data loss.
2. If SET SPILL OFF is in effect (the default), characters that have been shifted beyond the truncation column are truncated. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been shifted beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. SET SPILL affects only SHIFT issued with the RIGHT operand. SET SPILL has no effect on SHIFT LEFT; data shifted to the left past zone1 is lost.

Responses

If you specify a SHIFT is to occur on multiple lines and it does occur, the current line pointer:

1. Is unchanged, if SET STAY ON is in effect

2. Moves to the last line shifted, if SET STAY OFF is in effect (the default)

SHIFT does not affect the column pointer.

Messages and Return Codes

504E	*nn* **line(s) {truncated|spilled} [RC=3]**
520E	**Invalid operand:** *operand* **[RC=5]**
543E	**Invalid number:** *number* **[RC=5]**
545E	**Missing operand(s) [RC=5]**
546E	**Target not found [RC=2]**
585E	**No line(s) changed [RC=1 or 4]**

where return codes are:

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

SI (Macro)

```
►►──SI──────────────────────────────────────────────────────────────►◄
```

Purpose

Use the SI (Structured Input) macro to continuously add lines to a file. The first line is added following the line that contains the cursor. The cursor is then positioned at the column where the text on the preceding line begins. Each time you type on the new line and then press ENTER, another new line is automatically added to the file. If nothing is typed on the added line, the line is deleted when you press ENTER.

Usage Notes

1. SI can also be issued in the prefix area. See “SI (STRUCTURED INPUT) Macro” on page 440.
2. Using SI you can add a blank line to a file by making at least one change on the line. The line is considered changed if you press the space bar or if you retype the mask characters (see “SET MASK” on page 287). Moving the cursor using the cursor position keys over a line does not change the line.
3. After SI is terminated, the cursor is positioned at the indentation column of the last added line.
4. As lines are added using SI, any data above the new line remains stationary while the data below scrolls down the screen. When the new line is one line above the bottom of the file area, adding more lines scrolls the data above the new line up the screen. Using multiple SI subcommands or other prefix subcommands along with SI may move the new line off the screen. The cursor may not be placed at the indentation column when you issue multiple SI subcommands.

Notes for Macro Writers

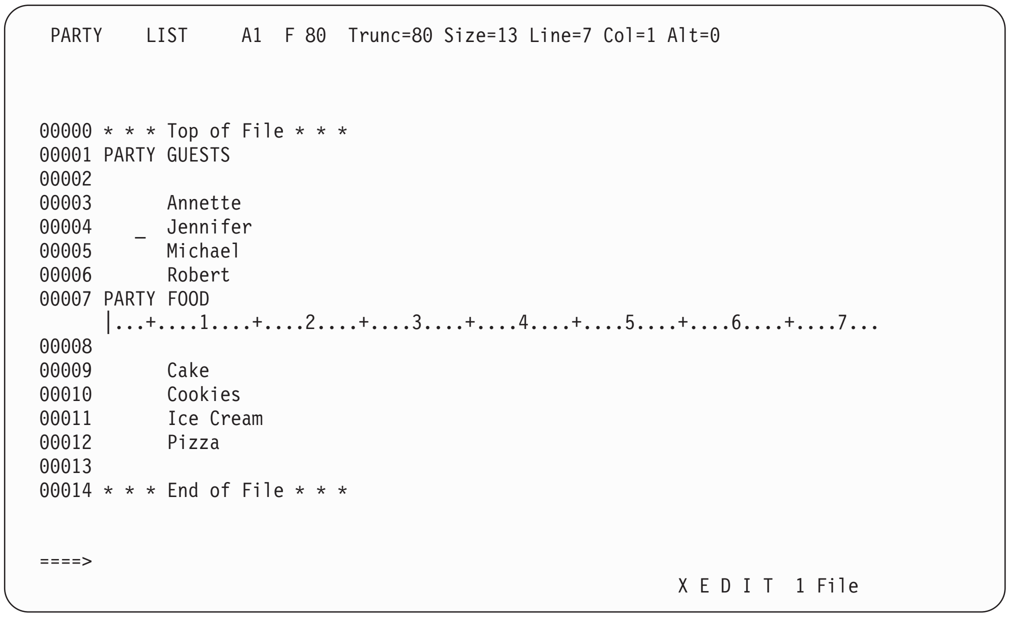
SI uses the console stack to adjust the current line when anything is pending on the line that will become the current line.

Examples

The following example shows how to use a PF key assigned as the SI subcommand.

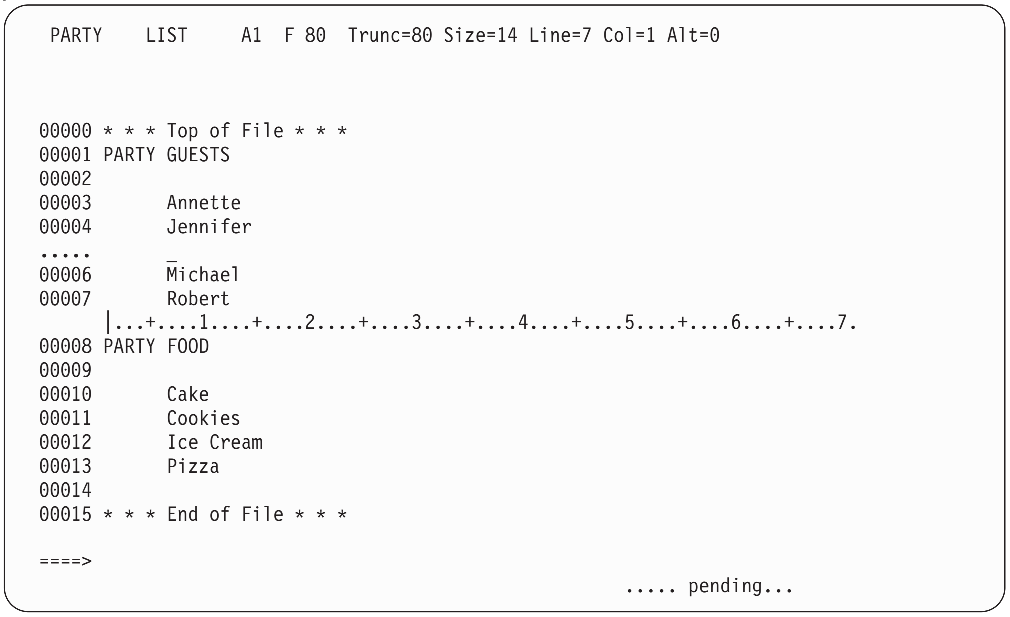
To insert the names of more party guests into the following file:

1. Move the cursor to the line where you want to add a name.
2. Press the PF key assigned to SI.



```
 PARTY    LIST     A1  F 80  Trunc=80 Size=13 Line=7 Col=1 Alt=0



00000 * * * Top of File * * *
00001 PARTY GUESTS
00002
00003       Annette
00004    _  Jennifer
00005       Michael
00006       Robert
00007 PARTY FOOD
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00008
00009       Cake
00010       Cookies
00011       Ice Cream
00012       Pizza
00013
00014 * * * End of File * * *


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

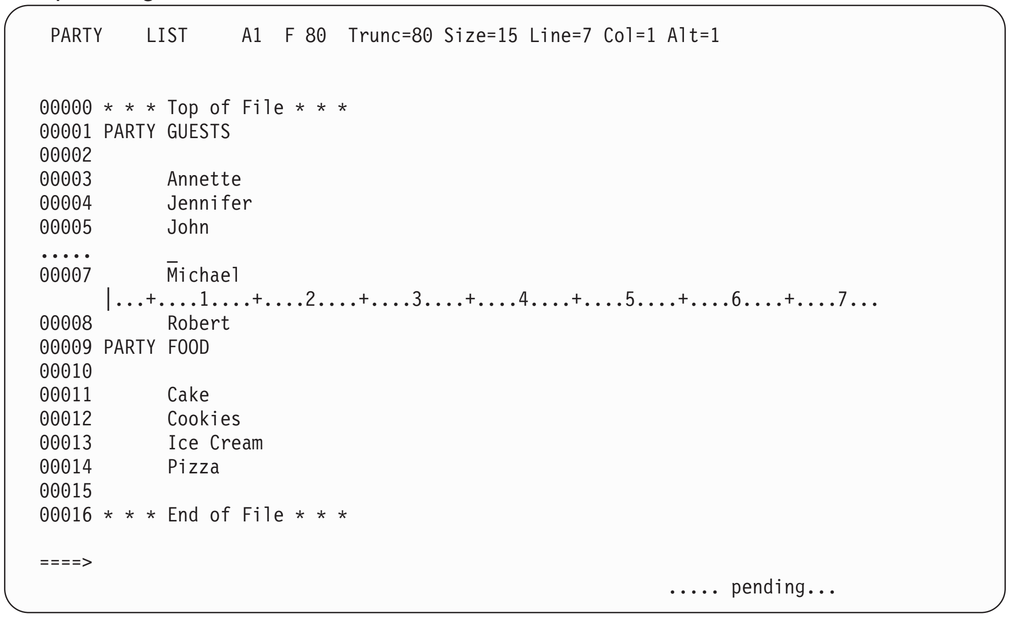
A new line is added after the line the cursor is on. The prefix area of the new line contains “. . . . .” and the cursor is positioned at the indentation column of the previous line.



```
 PARTY    LIST     A1  F 80  Trunc=80 Size=14 Line=7 Col=1 Alt=0



00000 * * * Top of File * * *
00001 PARTY GUESTS
00002
00003       Annette
00004       Jennifer
.....       _
00006       Michael
00007       Robert
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7.
00008 PARTY FOOD
00009
00010       Cake
00011       Cookies
00012       Ice Cream
00013       Pizza
00014
00015 * * * End of File * * *

====>
                                                       ..... pending...
```

Figure 17. SI Macro — Adding the First New Line

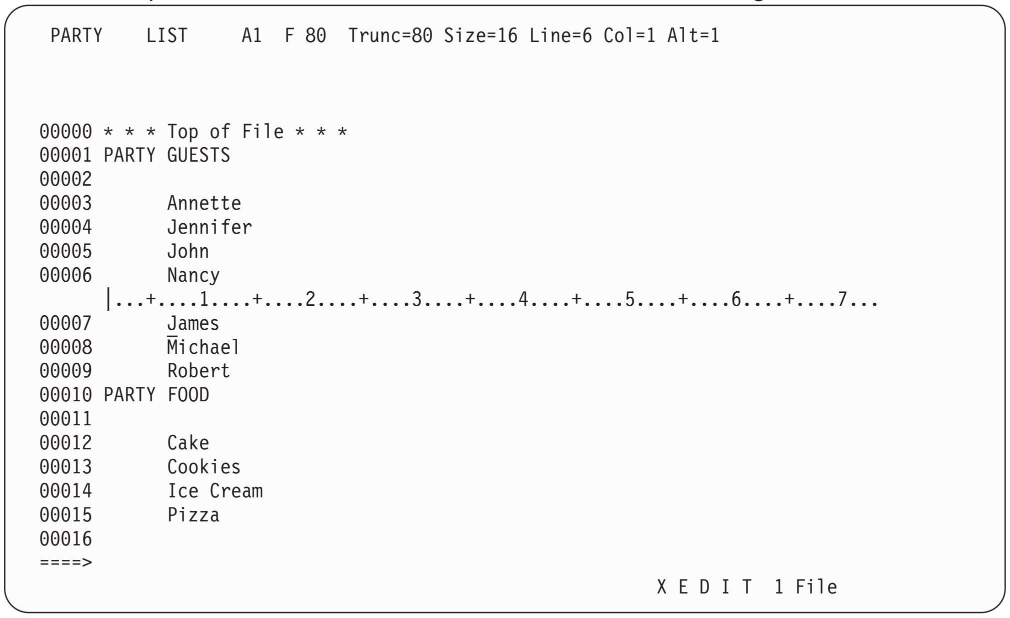
Type a name on the line and press ENTER. This adds a new line after the one you just typed on. You can continue to enter names to the list by typing on the new line and pressing ENTER.



```
 PARTY    LIST     A1  F 80  Trunc=80 Size=15 Line=7 Col=1 Alt=1


00000 * * * Top of File * * *
00001 PARTY GUESTS
00002
00003       Annette
00004       Jennifer
00005       John
.....       _
00007       Michael
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00008       Robert
00009 PARTY FOOD
00010
00011       Cake
00012       Cookies
00013       Ice Cream
00014       Pizza
00015
00016 * * * End of File * * *

====>
                                                  ..... pending...
```

Figure 18. SI Macro — Adds New Lines Until Null Line Entered (Part 1 of 2)

When you are finished adding names, press ENTER without typing on the new line. In this example two more names were added before terminating SI.



```
 PARTY    LIST     A1  F 80  Trunc=80 Size=16 Line=6 Col=1 Alt=1



00000 * * * Top of File * * *
00001 PARTY GUESTS
00002
00003       Annette
00004       Jennifer
00005       John
00006       Nancy
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00007       James
00008       Michael
00009       Robert
00010 PARTY FOOD
00011
00012       Cake
00013       Cookies
00014       Ice Cream
00015       Pizza
00016
====>
                                                 X E D I T  1 File
```

Figure 18. SI Macro — Adds New Lines Until Null Line Entered (Part 2 of 2)

Responses

The prefix area of the new line contains '. . . . .' and the following message is displayed in the status area:

```
..... pending...
```

This pending status allows you to continually add lines after you have typed on the new line.

By default, the prefix area for the added line is highlighted. For more information, see SET COLOR PENDING.

Each added line is prefilled with the current mask (see “SET MASK” on page 287).

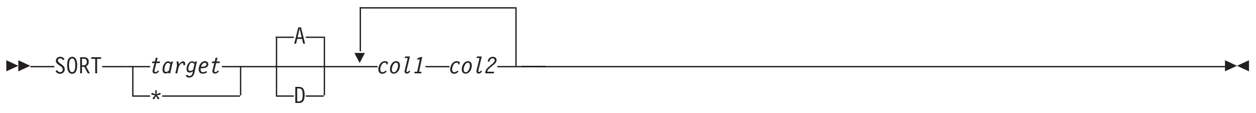
Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
529E SI is only valid in {display|editing} mode [RC=3]
561E Cursor is not on a valid data field [RC=1 or 3]

where return codes are:

3	Cursor is not in a valid field, or SI is valid in display mode only
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SORT (Macro)



Purpose

Use the SORT macro to arrange a specified number of file lines in ascending or descending EBCDIC order according to specified sort columns.

Operands

target
specifies the number of lines to be sorted. Lines are sorted starting with the current line, up to but not including the target line.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

***** lines are sorted from the current line to the end of the file.

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. The *col1* is the starting column of a sort field within each line and may not exceed 4096. The ending column is *col2*. You can specify as many sort fields as you want, as long as the total length does not exceed 248.

Usage Notes

1. If SET CASE UPPERCASE/MIXED RESPECT is in effect, sorting is done in EBCDIC order.
2. If SET CASE UPPERCASE/MIXED IGNORE is in effect, sorting is done in alphabetic order, with lowercase and uppercase representations of the same letter considered to be identical.
3. SORT operates outside the current SCOPE. (See Appendix B, "Effects of Selective Line Editing Subcommands," on page 455 for more information about SORT and the SCOPE setting.)
4. SORT cannot be issued when prefix subcommands or macros are pending and it cannot be issued from a prefix macro.
5. The SORT macro provides a stable sort. If lines have identical sort fields, their relative position in the file is maintained following the sort.

Examples

In this example, suppose you are editing a file that has two fields of information. The first field, positioned between columns 8 and 16 of the file, contains a list of file names. The second field, positioned between columns 17 and 24 of the file, contains a list of file types. Issuing the command:

```
sort * 17 24 8 16
```

sorts the file by file type (columns 17 through 24) and, within each file type, by file name (columns 8 through 16). The asterisk tells XEDIT to sort the entire file. Because no sort order is specified, the file is sorted in ascending order (the default).

Messages and Return Codes

009E	**Column *nn* exceeds record length [RC=24]**
053E	**Invalid sort field pair defined [RC=24]**
063E	**No sort list given [RC=40]**
493E	**SORT invalid in update mode [RC=3]**
545E	**Missing operand(s) [RC=5]**
546E	**Target not found [RC=2]**
554E	**Not enough virtual storage available [RC=104]**
581E	**Subcommand is not valid in extended mode [RC=3]**
588E	**Prefix subcommand waiting... [RC=8]**
596E	**This module must be called within the editor [RC=88]**

where return codes are:

0	Normal
1	TOF or EOF reached during execution
2	Target line not found
3	SORT cannot be used when a file is edited in UPDATE mode or extended mode
5	Missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
8	Prefix area contains pending subcommand or macro
24	Invalid columns defined
40	No list given
88	Module must be called within the editor
104	Insufficient storage available

SOS

```
►►──SOS───┬─Alarm─────────────┬──────────────────────────────────────────────────►◄
          ├─CLEAR─────────────┤
          ├─LINEAdd───────────┤
          ├─LINEDel───────────┤
          ├─NUlls────┬──────┬─┤
          │          ├─ON───┤ │
          │          └─OFF──┘ │
          ├─PFn───────────────┤
          ├─POP───────────────┤
          ├─PUsh──────────────┤
          │      ┌─1─┐        │
          ├─TABB─┴───┴────────┤
          │      └─n─┘        │
          ├─TABCmd────────────┤
          │          ┌─1─┐    │
          ├─TABCMDB──┴───┴────┤
          │          └─n─┘    │
          │          ┌─1─┐    │
          ├─TABCMDF──┴───┴────┤
          │          └─n─┘    │
          │      ┌─1─┐        │
          └─TABF─┴───┴────────┘
                 └─n─┘
```

Purpose

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.

Operands

ALarm
rings the terminal alarm the next time the display is refreshed.

CLEAR
clears the screen.

LINEAdd
adds a blank line after the line the cursor points to. This is the initial setting of the PF2 key.

LINEDel
deletes the line the cursor points to.

NUlls
reverses the current setting of the NULLS option for the line the cursor points to.

NUlls ON
changes the trailing blanks to nulls for the line the cursor points to.

NUlls OFF
changes the trailing nulls to blanks for the line the cursor points to.

PF*n*
depresses a PF key where *n* is the key number 1-24. The data that had been assigned to the key is placed LIFO in the CMS console stack.

POP
removes the top position from the cursor stack and places the cursor there. (See PUSH operand.) If the cursor is outside any logical screen, it is positioned in the upper left corner of the first logical screen.

PUsh
saves the current position of the cursor. The position is saved in a LIFO fashion in a five-position stack used only for this purpose.

TABB
TABB *n*
moves the cursor backward to the previous tab position as indicated in the SET TABS subcommand. The tab operation can be performed *n* times; 1 is the default.

TABCmd
positions the cursor at the command line for the logical screen which the cursor currently resides.

TABCMDB
TABCMDB *n*
moves 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; 1 is the default.

TABCMDF
TABCMDF *n*
moves 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; 1 is the default.

TABF
TABF *n*
moves the cursor forward to the next tab position as indicated in the SET TABS subcommand. The tab operation can be performed *n* times; 1 is the default.

Usage Notes

1. When the prefix is set on the left and the cursor is placed in the attribute byte following the line, TABF moves the cursor to the first tab column in the same line and TABB moves the cursor to the last tab column of the preceding line.
2. When using TABCMDF or TABCMDB, the view containing the first encountered command line becomes the file currently being edited. This switches editing to the next view displayed. As a result, if a TABCMDF or TABCMDB is issued when the commands are stacked (for example, by a PF key), the stacked commands are executed on the view containing the first encountered command line.
3. With ETMODE ON, TABF with the cursor on the command line can produce unpredictable results.
4. If you are in a GUI environment, the NULLS and ALARM operands are ignored.

Messages and Return Codes

520E Invalid operand: ***operand*** **[RC=5]**
529E Subcommand is only valid in {display|editing} mode [RC=3]
545E Missing operand(s) [RC=5]

561E Cursor is not on a valid data field [RC=3]

where return codes are:

0	Normal
1	Cursor is on TOF or EOF line TOF or EOF reached during execution
3	Invalid placement of cursor or subcommand, or subcommand is valid only for display terminal
5	Invalid or missing operand(s)
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SPLIT (Macro)

Format 1

```
                     ┌─Column─┐
►►──SPlit──┬─────────┬─┼────────┼──────────────────────────────────────►◄
           └─ALigned─┘ └─CURSOR─┘
```

Format 2

```
►►──SPlit──┬─────────┬─┬──────────────────────────────┬─────────────────►◄
           └─ALigned─┘ │ ┌────────────────────────────┐ │
                       │ ▼                            │ │
                       └───┬─colno──────────────────┬─┴─┘
                           │ ┌─Before─┐        (1)  │
                           └─┼────────┼─/string/────┘
                             └─After──┘
```

Notes:

1 The final delimiter (/) is optional after the last *string*. Leading or trailing blanks are considered part of *string*.

Purpose

Use the SPLIT macro to split a line into two or more lines.

Format 1 allows you to split a line into two lines, at the column pointer or at the cursor.

Format 2 allows you to split a line into several lines.

Operands

ALigned
gives the created line the same number of leading blanks as the original line.

Column
splits the current line into two lines. The second line starts with the data in the current column (as defined by the column pointer). The column pointer remains unchanged. If you enter SPLIT without an operand, SPLIT COLUMN is the default.

CURSOR
splits the line containing the cursor into two lines. The second line starts with the character under which the cursor was positioned. The cursor is not moved. This format of the SPLIT macro is especially useful when assigned to a PF key.

colno
splits the current line at the specified column number(s). The line is split as many times as there are operands.

Before
splits the current line *before* a specified character string. This is the default.

After
splits the current line *after* a specified character string.

/string/
splits the current line before or after the specified character string. The line is split as many times as there are operands.

Usage Notes

1. The SPLIT macro searches for strings in the current line with VARBLANK, SPAN, and STREAM all set to OFF (and restored after the SPLIT).
2. SPLIT is the converse of JOIN. (See “SPLTJOIN (Macro)” on page 388, which combines the functions of SPLIT and JOIN.)
3. The cursor or the column number or string specified must fall within the current zones.

Examples

This section shows examples of the SET SPLIT subcommand.

Example 1

Current Line:

```
===== Electric eels can discharge bursts of 625 volts.
```

Note the cursor position under the b in bursts. Press a PF key assigned to SPLIT CURSOR.

```
===== Electric eels can discharge _
===== bursts of 625 volts.
```

Example 2

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.
```

Example 3

Current Line:

```
=====    Electric eels can discharge bursts of 625 volts.
```

`sp aligned 23` (Split at column 23, but line up with first line.)

```
=====    Electric eels can
=====    discharge bursts of 625 volts.
```

Example 4

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.
```

Example 5

Current Line:

```
===== Electric eels can discharge bursts of 625 volts.
```

```
sp 15 b/bursts/
```
(Split the line into three lines.)

```
===== Electric eels
===== can discharge
===== bursts of 625 volts.
```

The new current line is the last one that was added as a result of the split.

Responses

The second line begins in column 1 unless SET IMAGE ON is in effect, in which case the second line begins in the first tab column.

If the current line is split, the last line added becomes the new current line. If SPLIT CURSOR is issued, the line pointer remains unchanged.

If a specified string is not found, an error message is issued and the current line remains the same.

Messages and Return Codes

526E	**Option** *option* **valid in display mode only [RC=3]**
557S	**No more storage to insert lines [RC=4]**
561E	**Cursor is not on a valid data field [RC=1 or 3]**
575E	**Invalid [argument or] {JOIN|SPLIT|TABS|VERIFY|ZONE} column(s) defined [RC=5]**
585E	**No line(s) changed [RC=1 or 4]**
586E	**Not found [RC=2]**

where return codes are:

0	Normal
1	No change (SPLIT issued at TOF or EOF)
2	Not found
3	Invalid placement of cursor or subcommand, or subcommand is valid only for display terminal
4	No more storage
5	Invalid operands
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

SPLTJOIN (Macro)

```
►►──SPLTJOIN────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the SPLTJOIN macro either to split a line or join two lines, depending on the position of the cursor on a file line. If the cursor is positioned before or at the last nonblank character, the line is split (at the cursor position). If the cursor is positioned after the last nonblank character on a line (that is, after the end of the data on a line), the next line is appended, starting at the cursor position.

Usage Notes

1. The SPLTJOIN macro is most useful when assigned to a PF key. The PF11 key is initially set to SPLTJOIN.
2. SPLTJOIN enables display terminal users to combine the functions of SPLIT CURSOR and JOIN CURSOR (see "SPLIT (Macro)" on page 385 and "JOIN (Macro)" on page 135) on a single PF key. Unlike JOIN CURSOR, it prevents the accidental loss of data caused by joining two lines and overlaying data.
3. SPLTJOIN functions like SPLIT and JOIN issued with the ALIGNED operand, that is, it *takes care of* leading blanks. If a line is split, it adds the same number of leading blanks to the beginning of the new line as the original line has. If two lines are joined, it removes the same number of leading blanks from the line being joined as there are on the line to which it is appended.
4. The cursor must lie within the current zones.
5. If SET SPILL ON or SET SPILL WORD is in effect, characters that have been pushed beyond the truncation column are inserted in the file as one or more new lines, starting with the first character or word that would have gone beyond the truncation column. For the SPLTJOIN macro, SET SPILL OFF (the default) has the same effect as SET SPILL ON. SPLTJOIN does not truncate data.
6. If two lines are joined, the original line appended as a result of the join is deleted. (The line pointer remains unchanged.)

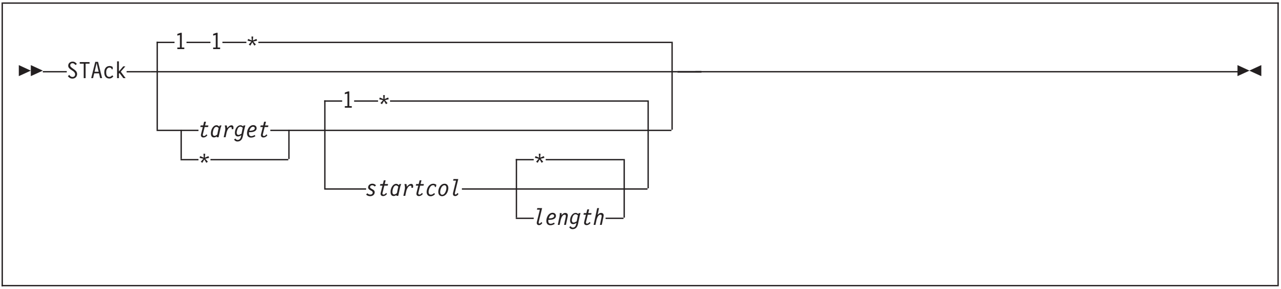
Messages and Return Codes

503E	**{Truncated|Spilled} [RC=3]**
520E	**Invalid operand:** *operand* **[RC=5]**
529E	**SPLTJOIN is only valid in {display|editing} mode [RC=3]**
557S	**No more storage to insert lines [RC=4]**
561E	**Cursor is not on a valid data field [RC=1 or 3]**
585E	**No line(s) changed [RC=1 or 4]**

where return codes are:

0	Normal
1	No change (SPLTJOIN issued at TOF or EOF)
3	Spill occurred, or invalid placement of cursor or subcommand, or subcommand is valid only for display terminal
4	No more storage
5	Invalid operand
6	Subcommand rejected in the PROFILE due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring

STACK



Purpose

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.

Operands

target
defines the number of lines to be stacked. Lines are stacked starting with the current line, up to but not including the target line. If *target* is not specified, only the current line is stacked.

You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

***** the rest of the file is stacked.

startcol
specifies lines are to be stacked starting in this column number. If *startcol* is not specified, the line is stacked starting at the first column.

length
specifies the number of columns to be stacked, starting with *startcol*. The maximum value for *length* is the truncation column. If *length* is not specified, the line is stacked up to the truncation column.

Notes for Macro Writers

1. STACK 1 1 0 stacks an empty line. STACK 0 also stacks an empty line.
2. The CMS console stack restricts the *length* to be stacked to 255 bytes. To retrieve information about lines longer than 255 bytes, use EXTRACT/CURLINE/ in a REXX macro.
3. The line pointer is moved to the last line stacked.
4. If you specify a backward target (for example, –3) lines are stacked in reverse order.

Messages and Return Codes

520E Invalid operand: *operand* **[RC=5]**
543E Invalid number: *number* **[RC=5]**
546E Target not found [RC=2]

where return codes are:

0	Normal

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

STATUS (Macro)

```
►►──STATus──┬────┬────────────────────────────────────────────────────────►◄
            └─fn─┘
```

Purpose

Use the STATUS macro to display the SET subcommand options and their current settings, or to create an XEDIT macro that contains the SET subcommand options listed under Usage Note 2.

Operands

fn specifies the name of a file that is to contain all the SET subcommand options listed in Usage Note 2.

The editor assigns a file type of XEDIT; therefore, the file is an XEDIT macro. The macro can be invoked later to restore the SET subcommand options that were in effect when the STATUS macro was issued.

Usage Notes

1. Use the STATUS macro without an operand to display the SET subcommand options and their current settings. All SET subcommand options are displayed except:

   ```
   ALT       PF
   BRKKEY    POINT
   ENTER     RESERVED
   ETARBCH   SELECT
   LASTLORC  SIDCODE
   PA        TRANSLAT
   PENDING   =
   ```

   If you are in a GUI environment, the response is displayed in the XEDIT Messages window.
2. When you use the STATUS macro to create an XEDIT macro, you can later invoke the macro to restore the SET subcommand options that were in effect when the STATUS macro was issued.

   For example:

   ```
   status KEEP
   ```

   KEEP is the name of the macro.

   Settings of the following subcommands are included in the macro:

   ```
   APL             FNAME           NUMBER          STREAM
   ARBCHAR         FTYPE           PACK            SYNONYM
   AUTOSAVE        FULLREAD        PNAME           TABLINE
   BFSLINE         HEX             PREFIX          TABS
   CASE            IMAGE           RANGE           TERMINAL
   CMDLINE         IMPCMSCP        RECFM           TEXT
   COLOR           LINEND          REMOTE          TOFEOF
   COLPTR          LRECL           SCALE           TRUNC
   CTLCHAR         MACRO           SCOPE           VARBLANK
   CURLINE         MASK            SCREEN          VERIFY
   DISPLAY         MSGLINE         SERIAL          WRAP
   ESCAPE          MSGMODE         SHADOW          ZONE
   ETMODE          NAMETYPE        SPAN
   FILLER          NONDISP         SPILL
   FMODE           NULLS           STAY
   ```

3. With the exception of SET PNAME, an XEDIT macro created with the STATUS macro cannot be used to restore the SET subcommand options if any of the SET subcommands were issued containing one or more single quotes ('). For example:
   ```
   set ctlchar '
   ```

Responses

```
703I File file name XEDIT A1 created
```

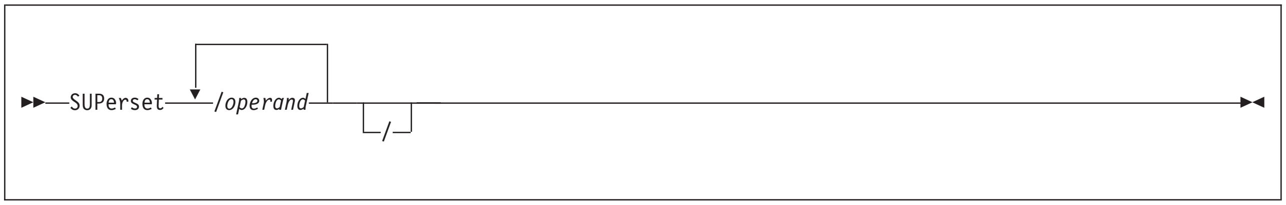
Messages and Return Codes

024E File *fn* **XEDIT A already exists [RC=28]**
520E Invalid operand: *operand* **[RC=5]**
671E Error {sending|receiving|creating|loading|updating} [file] *fn ft fm***, rc=***nn* **from EXECIO [RC=31, 55, 70, 76, 99, 100]**

where return codes are:

0	Normal
5	Invalid operand
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
28	File name specified already exists
31	A rollback occurred
55	APPC/VM communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
99	A required system resource is not available
100	Error occurred while creating the file

SUPERSET



Purpose

Use the SUPERSET subcommand to enter multiple SET options on one subcommand to improve performance.

Operands

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

operand
See “SET” on page 234 or enter HELP SET while in XEDIT mode for the existing options and additional usage notes available with the SUPERSET subcommand.

Usage Notes

1. The following will assist you in combining multiple subcommands with SUPERSET.

   SET ETMODE
   The delimiter chosen should be different from the SO character (X'0E').

   SET LINEND
   The delimiter chosen should be different from the LINEND character.

   SET MASK
   If you use SET MASK MODIFY or SET MASK DEFINE in a macro, SUPERSET will stop processing, pause for input, and then continue.
2. SUPERSET will continue if the individual SET subcommands return with RC=0. The only exception to this is SET RANGE, which can return with RC=1 (EOF reached).
3. The maximum length of an XEDIT subcommand from a macro, after variable expansion, is limited to 256 characters. Use caution when determining how many SET subcommands can be combined into one SUPERSET subcommand.

Examples

The following is an example of how you can use SUPERSET with any of the SET options to improve performance.

```
superset /scale on/autosave 5/case mixed/tabs 5 10/
```

quickly sets your scale on, issues the SAVE subcommand after 5 changes to the file, lets you type in mixed case, and sets tabs at column 5 and column 10.

Messages and Return Codes

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

where return codes are:

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

TOP

```
►►──TOP──────────────────────────────────────────────────────────────────────►◄
```

Purpose

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 “SET RANGE” on page 317).

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, and so forth). If the current line is the first line of the file, a string occurring in this line would be missed, because LOCATE, FIND, and so forth start searching with the line *following* the current line.

Responses

The null line at the top of the file becomes the new current line and contains:

```
* * * Top of File * * *
```

The lines preceding the current line are blank, and the rest of the screen contains the beginning lines of the file.

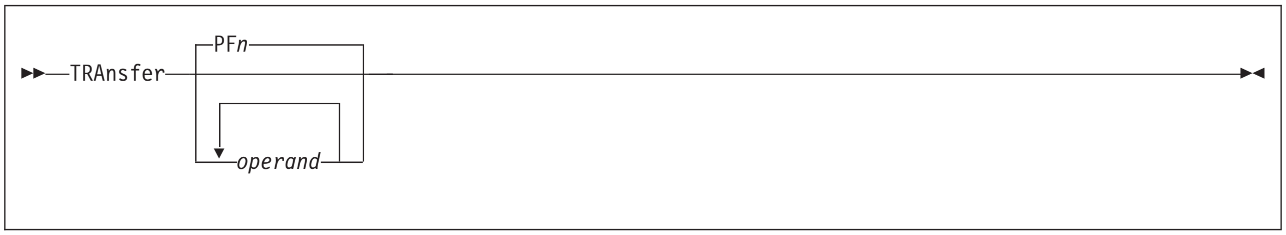
Messages and Return Codes

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

where return codes are:

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

TRANSFER



Purpose

The TRANSFER subcommand is used *within a macro* to access specified editing variables, for example, the current line number, the file name of the file being edited, and so forth. The values transferred are placed in the console stack, and subsequent EXEC 2 &READ control statements can read them. You can specify more than one keyword in one TRANSFER subcommand, placing the values last-in-first-out (LIFO) on the console stack.

Operands

operand
will be any one of the following keywords. For additional information on some of the operands, refer to its corresponding SET subcommand (if applicable).

APL	LRecl	STAY
ARBchar	LScreen	STReam
AUtosave	MACRO	SYNonym
CASE	MASK	TABLine
CMDline	MSGMode	TABS
COLPtr	NBFile	TARGet
COLumn	NONDisp	TERMinal
CTLchar	NULls	TEXT
CURLine	NUMber	TOF
CURSor	PACK	TOFEOF
EOF	PF*n*	TRunc
ESCape	Point	UPDate
FILler	PREfix	VARblank
FMode	RANge	Verify
FName	RECFm	VERShift
FType	RESERved	Width
HEX	SCALe	WRap
IMage	SCReen	Zone
IMPcmscp	Seq8	=
LASTmsg	SERial	
LENgth	SIDcode	
LIne	SIZe	
LINENd	SPAN	

APL
transfers ON or OFF as defined in the SET APL subcommand.

ARBchar
transfers ON or OFF and the arbitrary character specified in the SET ARBCHAR subcommand.

AUtosave
transfers the current AUTOSAVE setting: the AUTOSAVE count, file ID, and number of alterations.

CASE
transfers two values: the case setting (U or M) and R or I as defined in the SET CASE subcommand.

CMDline
transfers an integer (*n*), which is the screen command line number defined with the SET CMDLINE subcommand. If SET CMDLINE TOP, *n*=2. If SET CMDLINE ON, the number is the logical screen size minus one. If SET CMDLINE BOTTOM, *n* is the number of the last line on the logical screen. If SET CMDLINE OFF, *n*=0.

COLPtr
transfers ON or OFF as defined in the SET COLPTR subcommand.

COLumn
transfers the column number of the column pointer.

CTLchar

```
►►──TRAnsfer──CTLchar──┬──────┬──────────────────────────────────────►◄
                       └─char─┘
```

With *char* omitted (TRAnsfer CTLCHAR), transfers the control character identifier and all characters defined in the SET CTLCHAR subcommand, in the form CTLCHAR ON ESCAPE *char* CTL *c1 c2 c3 c4* and so forth. If no control characters are defined, transfers CTLCHAR OFF.

With *char* specified (TRAnsfer CTLCHAR *char*), the attributes of that character are transferred, in the form CTLCHAR *char* attribute1 PROTECT|NOPROTECT attribute2 HIGH|NOHIGH|INVISIBLE. If no attributes were defined for the character, transfers CTLCHAR.

If the TRANSFER subcommand specifies multiple operands, the operand that follows CTLCHAR is interpreted as follows: if it is one character in length, it is interpreted as the *char* operand of CTLCHAR; if it is longer than one character or is not specified, it is handled normally.

For example:

```
transfer fn ft fm ctlchar lrecl recfm
```

CTLCHAR is treated as if it were specified without the *char* operand. LRECL and RECFM are handled normally.

```
transfer ctlchar ¢ ctlchar " ctlchar % lrecl recfm
```

transfers the attributes of ¢, “, and %.

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

CURSor
transfers four integers: the current position of the cursor on the screen (line number and column number) and the position of the cursor in the file (line number and column number). If the cursor is in a protected area, two negative numbers (–1) are transferred for the position of the cursor in the file. The top and bottom of the range are considered to be in the file.

The current position of the cursor is the location where the cursor would be placed if the screen were displayed at this time.

EOF
transfers ON or OFF as the editor determines. EOF is ON when the line pointer reaches end of file.

ESCape
transfers ON or OFF and the escape character (one-character string) defined in the SET ESCAPE subcommand. This character may be blank.

FILler
transfers the filler character (one-character string) defined in the SET FILLER subcommand. This character may be blank.

FMode
transfers the two-character file mode.

FName
transfers the eight-character file name.

FType
transfers the eight-character file type.

HEX
transfers ON or OFF as specified in the SET HEX subcommand.

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

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

LASTmsg
transfers the last message the editor issued. This message may or may not have been displayed, depending on the SET MSGMODE subcommand operands.

LENgth
transfers the length of the current line from column one through the truncation column, excluding trailing blanks.

LIne
transfers the current line number, relative to the beginning of the file.

LINENd
transfers ON or OFF and the line-end character as defined in the SET LINEND subcommand.

LRecl
transfers the logical record length.

LScreen
transfers six integers: the number of lines and the number of columns of the logical screen; the line number and column number defining the top left corner of the logical screen on the virtual screen; the number of lines and number of columns of the virtual screen.

MACRO
: transfers ON or OFF as specified in the SET MACRO subcommand.

MASK
: transfers the current mask line as defined in the SET MASK subcommand. The line may be all blanks.

MSGMode
: transfers ON or OFF and LONG or SHORT as defined in the SET MSGMODE subcommand.

NBFile
: transfers the number of files being edited.

NONDisp
: transfers the character defined in the SET NONDISP subcommand. The character may be blank.

NULls
: transfers ON or OFF as specified in the SET NULLS subcommand.

NUMber
: transfers ON or OFF as specified in the SET NUMBER subcommand.

PACK
: transfers ON or OFF as specified in the SET PACK subcommand.

PF*n*
: transfers the string associated with a specified PF key, as defined in SET PF*n*. The string can be null or blank.

Point
: transfers the symbolic name associated with the current line, as defined in the SET POINT subcommand or the .xxxx prefix subcommand, or transfers a blank string if no name has been defined.

PREfix
: transfers ON, OFF, or NULLS and RIGHT or LEFT as specified in the SET PREFIX subcommand.

RANge
: transfers two integers, which are the line numbers of the top and bottom of the range defined in the SET RANGE subcommand.

RECFm
: transfers the record format, F, V, FP, or VP, defined in the SET RECFM subcommand.

RESERved
: transfers as one line the line numbers of reserved lines.

SCALe
: transfers ON or OFF and the scale line number as specified in the SET SCALE subcommand.

SCReen
: transfers SIZE *n1 n2*. . ., where *n1* is the number of lines in the first logical screen, *n2* is the number of lines in the second logical screen, and so forth, as defined in the SET SCREEN subcommand.

Seq8
: transfers OFF if the XEDIT command was issued with the NOSEQ8 operand; if not, transfers ON.

SERial
transfers the serial identification or OFF, the increment value, and the serial number starting value as defined in 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

```
►►──TRAnsfer──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 after SYNONYM is 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 in 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 the editor determines. TOF is ON when the current line pointer reaches the top of file.

TOFEOF
transfers ON or OFF as specified in the SET TOFEOF subcommand.

TRunc
transfers the truncation column number as defined in the SET TRUNC subcommand.

UPDate
transfers ON or OFF as the editor determines. Update is ON when the XEDIT command has been issued with the UPDATE or CTL operands.

VARblank
transfers ON or OFF as specified in the SET VARBLANK subcommand.

Verify
transfers ON or OFF, H (if SET VERIFY with the HEX option was previously issued), and the verification columns as specified in the SET VERIFY subcommand.

VERShift
transfers +*n* or –*n*, which is the relative position of the screen over the file as a result of any LEFT or RIGHT subcommands.

Width
transfers the WIDTH value as specified in the XEDIT command or LOAD subcommand or as the editor determines.

WRap
transfers ON or OFF as specified in the SET WRAP subcommand.

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

= transfers one line (up to 130 characters) that corresponds to the content of the *equal (=) buffer.* The equal buffer contains the last executed subcommand or macro or CP/CMS command, or whatever has been specified in the SET = subcommand.

Notes for Macro Writers

1. One TRANSFER subcommand can transfer several values. For example:
   ```
   transfer line size trunc
   ```
   The macro can then read the values. For example:
   ```
   &read vars &line &size &trunc
   ```
   The variables in the macro now contain the current line number, the size of the file, and the truncation column, respectively.
2. Remember some TRANSFER keywords are associated with a *set* of values (like CURSOR and TABS), or a character that can 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.

Messages and Return Codes

545E Missing operand(s) [RC=5]

where return codes are:

0	Normal
5	Missing operand(s)

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

TYPE

```
          ┌─1──────┐
►►──Type──┼────────┼──────────────────────────────────────────────────►◄
          ├─target─┤
          └─*──────┘
```

Purpose

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

Operands

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 *target* is not specified, only the current line is displayed.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see "LOCATE" on page 147 and *z/VM: XEDIT User's Guide*.

* the rest of the file is displayed.

Usage Notes

1. TYPE displays a line according to the current SET VERIFY subcommand.
2. TYPE displays the first 256 characters for each line in the message area. If the output exceeds the number of lines for the message line, the output is passed to CMS. When full-screen CMS is ON, the output appears in the CMSOUT window. To see all the information in the virtual screen, you can use the CMS WINDOW FORWARD or WINDOW BACKWARD command. The screen is cleared automatically when you scroll to the bottom of the virtual screen. Alternately, you can clear the screen with the CMS WINDOW DROP command. When full-screen CMS is OFF, press CLEAR to redisplay the file.

   If you are in a GUI environment, messages that do not fit in the message area are displayed in the XEDIT Messages window.
3. If SET SHADOW ON and SET SCOPE DISPLAY are in effect, any shadow lines within the lines being typed are also displayed:

   ```
   ------------- nn line(s) not displayed --------------
   ```

   However, shadow lines are not included in the count of lines to be typed.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Responses

The specified lines are displayed.

The line pointer moves to the last line typed.

If the line pointer has moved to the null End of File line, the last line displayed is:

```
583I   EOF:
```

If the current line is the null Top of File and you issue the TYPE subcommand, the first line displayed is:

```
584I   TOF:
```

Messages and Return Codes

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

where return codes are:

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

UP

```
        ┌─1─┐
►►──Up──┼───┼──────────────────────────────────────────────────────►◄
        ├─n─┤
        └─*─┘
```

Purpose

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

Operands

n is the number of lines the line pointer is to be moved toward the top of the file. If *n* is not specified, the pointer is moved up only one line.

* the line pointer moves to the Top of File line.

Usage Notes

1. The UP subcommand is equivalent to a minus (–) target. For example:

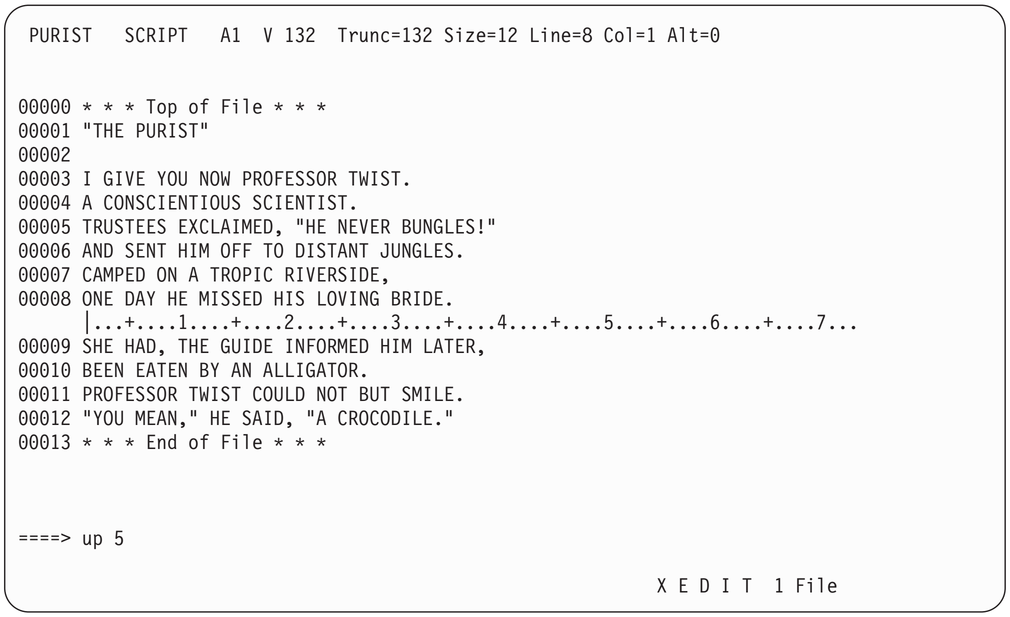
   ```
   up 3
   ```

   is equivalent to

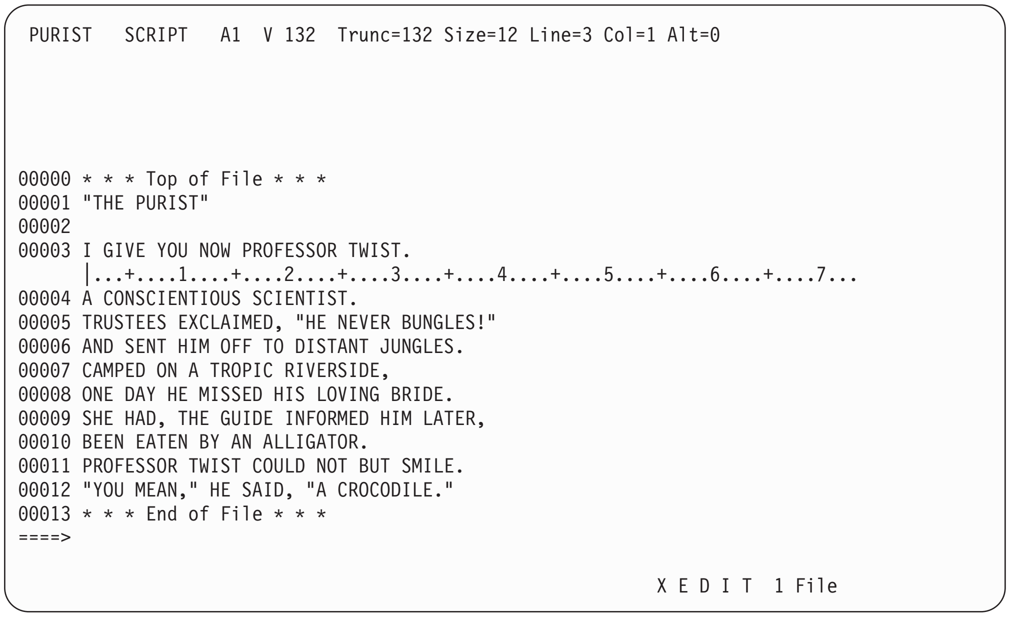
   ```
   -3
   ```

Examples

Figure 19 on page 406 is a before-and-after example of the UP subcommand.



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=8 Col=1 Alt=0


00000 * * * Top of File * * *
00001 "THE PURIST"
00002
00003 I GIVE YOU NOW PROFESSOR TWIST.
00004 A CONSCIENTIOUS SCIENTIST.
00005 TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
00006 AND SENT HIM OFF TO DISTANT JUNGLES.
00007 CAMPED ON A TROPIC RIVERSIDE,
00008 ONE DAY HE MISSED HIS LOVING BRIDE.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00009 SHE HAD, THE GUIDE INFORMED HIM LATER,
00010 BEEN EATEN BY AN ALLIGATOR.
00011 PROFESSOR TWIST COULD NOT BUT SMILE.
00012 "YOU MEAN," HE SAID, "A CROCODILE."
00013 * * * End of File * * *



====> up 5
                                                             X E D I T  1 File
```



```
 PURIST   SCRIPT   A1  V 132  Trunc=132 Size=12 Line=3 Col=1 Alt=0





00000 * * * Top of File * * *
00001 "THE PURIST"
00002
00003 I GIVE YOU NOW PROFESSOR TWIST.
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
00004 A CONSCIENTIOUS SCIENTIST.
00005 TRUSTEES EXCLAIMED, "HE NEVER BUNGLES!"
00006 AND SENT HIM OFF TO DISTANT JUNGLES.
00007 CAMPED ON A TROPIC RIVERSIDE,
00008 ONE DAY HE MISSED HIS LOVING BRIDE.
00009 SHE HAD, THE GUIDE INFORMED HIM LATER,
00010 BEEN EATEN BY AN ALLIGATOR.
00011 PROFESSOR TWIST COULD NOT BUT SMILE.
00012 "YOU MEAN," HE SAID, "A CROCODILE."
00013 * * * End of File * * *
====>

                                                             X E D I T  1 File
```

Figure 19. UP Subcommand — Before and After

Responses

The line pointed to becomes the new current line.

Messages and Return Codes

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

where return codes are:

0	Normal
1	Top of File reached and displayed
5	Invalid operand or number

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

UPPERCAS

```
           ┌─1──────┐
►►──UPPercas──┼────────┼──────────────────────────────────────────────────►◄
           ├─target─┤
           └─*──────┘
```

Purpose

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

Operands

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 *target* is not specified, only the current line is translated.

  You can specify a target as an absolute line number, a relative displacement from the current line, a line name, or a string expression. For more information on targets, see “LOCATE” on page 147 and *z/VM: XEDIT User’s Guide*.

* the rest of the file is translated.

Usage Notes

1. UPPERCAS does not change the case setting defined in the SET CASE subcommand.

Examples

In the following example, the current line is translated to uppercase. For more information, see *z/VM: XEDIT User’s Guide*.

Current Line:

```
===== Elephant tusks can weigh up to 300 pounds.

upp

===== ELEPHANT TUSKS CAN WEIGH UP TO 300 POUNDS.
```

Responses

1. When you press ENTER, all lowercase letters within the current zones appear in uppercase.
2. If you specify that UPPERCAS is to occur on multiple lines and it does occur, the current line pointer:
   a. Is unchanged if SET STAY ON is in effect
   b. Moves to the last line translated if SET STAY OFF is in effect (the default)

Messages and Return Codes

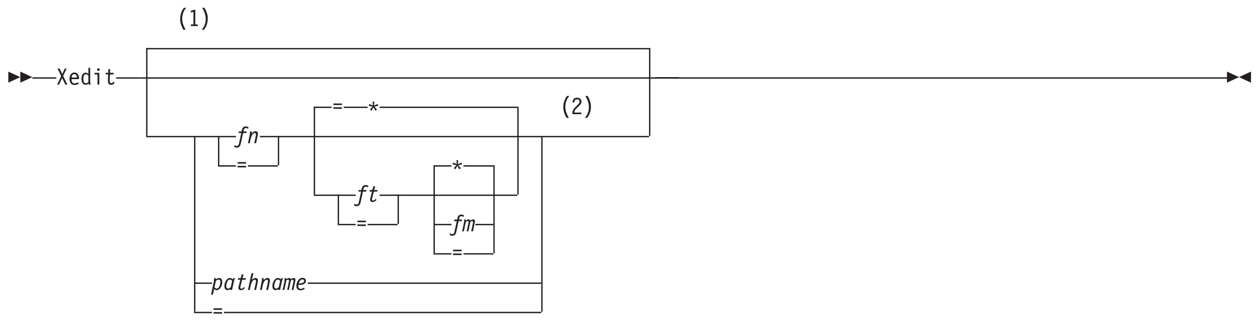
520E Invalid operand: *operand* **[RC=5]**
546E Target not found [RC=2]

585E No line(s) changed [RC=4]

where return codes are:

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

XEDIT



Notes:

1 If operands are not specified, the next file in the ring is chosen.

2 See the XEDIT command for the list of operations.

Purpose

Use the XEDIT subcommand to edit multiple files.

Operands

fn is the file name of a file to be brought into virtual storage. If the file is already in storage, no new copy is brought in.

ft is the file type of the above file. If you omit *ft*, the editor uses the file type of the file you are currently editing.

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

- Editing existing files

  When you specify the file mode, that disk or directory and its extensions are searched. If you omit the file mode or specify it as an asterisk (*), if the file is not already in the ring, all accessed disks and directories are searched for the specified file.

- Creating new files

  If you omit the file mode, the editor assumes a file mode of A1.

pathname
is the name of the byte file system (BFS) file to be edited. See “Understanding Byte File System (BFS) Path Name Syntax” on page 4 for a description of the different forms of the BFS path name syntax.

Options
are the same as the options and update mode options in the CMS XEDIT command. (See Chapter 2, “The XEDIT Command,” on page 11.)

Usage Notes

1. The XEDIT subcommand allows you to independently edit multiple files in virtual storage. To edit another file in addition to the ones you are already editing, use the XEDIT subcommand and specify the new file ID.

The files are placed in a *ring*. Each time you issue the XEDIT subcommand without arguments, the next file in the ring appears on the screen as the current file. This arrangement lets you switch from the first file to the second, the second to the third, and so forth, all the way around the ring and back to the first.

Each time you enter the XEDIT subcommand with a file ID, the file name and file type (or path name) specified are compared to all files in the ring to determine if this file is already being edited. If a match is found, that file is made the current file and any options specified are ignored. If a match is not found, the specified file is brought into storage and becomes the current one.

Issuing a QUIT or FILE subcommand from a file displays the previous file in the ring.

2. XEDIT in member mode creates a file with a file type of MEMBER. The resulting file may have a file ID identical with a file already in the ring.
3. XEDIT in update mode creates a file with a file type of UPDATE or the file type of the last update file applied, if any.
4. By changing the fm of a file (to ***S***, for example), you can then edit another file with a file ID identical to the original file in the same ring.
5. When the screen is divided into multiple logical screens (see "SET SCREEN" on page 329), entering the XEDIT subcommand with the same file ID from two (or more) logical screens creates two (or more) independent views of the same file.
6. When NAMETYPE CMS is specified or defaults, specifying an equal sign (=) for the *fn, ft,* and *fm* operands causes the corresponding FNAME, FTYPE, and FMODE values to be used. If not specified as an equal sign, the operand will be translated to uppercase.

   If you are in an XEDIT session when NAMETYPE is BFS, and you edit a CMS file by using the '=' without setting FNAME and FTYPE, the editor returns an error if you attempt to FILE or SAVE the CMS file. If you specify '=' for file mode without setting FMODE, it defaults to file mode A.

   When the NAMETYPE BFS option is used, specifying an equal sign (=) for the *pathname* operand causes the corresponding PNAME value to be used. The path name is converted to uppercase only when SET CASE UPPER is in effect.

   If you are in an XEDIT session when NAMETYPE is CMS, and you edit a BFS file by using the '=' without setting PNAME, the editor returns an error if you attempt to FILE or SAVE the BFS file.
7. Commit behavior of BFS files is different from that of SFS or minidisk files. For example, if the editor encounters an error when writing to a minidisk or SFS file, your changes are reversed and the original file is preserved. This is not the case for BFS files. The editor creates a copy of the file in XEDTEMP CMSUT1 on your A disk so you can recover your data if an error occurs when writing the file to the byte file system.

   If the editor is unable to create the XEDTEMP CMSUT1 file, you will receive a message describing the problem, and a second message:

   ```
   595E  Not able to create CMS file used for recovery.  Correct
         error or QQUIT to exit without writing file
   ```

   If the editor encounters an error while writing the byte file system file such that the contents of an existing file are damaged, you will receive a message describing the problem, and a second message to tell you that you must take action to recover your file.

   ```
   024E  File XEDTEMP CMSUT1 A1 contains file contents; use
         OPENVM PUT to recover file
   ```

8. If this subcommand is issued as “CMS XEDIT. . .”, XEDIT is called recursively. (See page 18.)
9. If you XEDIT an SFS file that has been migrated (moved to DFSMS/VM®-owned storage), it is automatically recalled if the CMS SET RECALL command is set to ON. (See *z/VM: CMS Commands and Utilities Reference* for an explanation of this command.) This may cause a slight delay before the file can be accessed.
10. You can XEDIT path names only if they represent BFS regular files.
11. The WINDOW, NOCLEAR, and NOSCREEN options are ignored if you are in a GUI environment.

Responses

Responses are the same as in the CMS XEDIT Command. (See Chapter 2, “The XEDIT Command,” on page 11.)

Messages and Return Codes

002E	**File** *fn ft fm* **not found [RC=28]**
003E	**Invalid option:** *option* **[RC=24]**
024E	**File XEDTEMP CMSUT1** *fm* **already exists [RC=28]**
029E	**Invalid parameter** *parameter* **in the option** *option* **field [RC=24]**
033E	**File is not a regular BFS file [RC=32]**
048E	**Invalid filemode** *mode* **[RC=24]**
054E	**Incomplete [or incorrect] fileid specified [RC=24]**
062E	**Invalid character in fileid {***fn ft fm***|***pathname***} [RC=20]**
065E	*Option* **option specified twice [RC=24]**
066E	*Option1* **and** *option2* **are conflicting options [RC=24]**
069E	**Filemode** *mode* **not accessed [RC=36]**
070E	**Invalid parameter** *parameter* **[RC=24]**
104S	**Error** *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 55, or 100]**
109S	**Virtual storage capacity exceeded [RC=104]**
132S	**File [***fn ft fm***] too large[:** *pathname***] [RC=88]**
137S	**Error** *nn* **on STATE for** *fn ft fm* **[RC=88]**
229E	**Unsupported OS dataset, error** *nn* **[RC=80, 81, 82, or 83]**
500E	**Unable to unpack file** *fn ft fm* **[RC=88]**
508E	**LOAD must be the first subcommand in the profile [RC=3]**
512E	**This is not allowed in CMS subset mode [RC=100]**
554E	**Not enough virtual storage available [RC=104]**
556I	**Editing existing empty file:**
571I	**Creating new file:**
622E	**Insufficient free storage [for {MSGLINE|PFkey/PAkey|synonyms}]**
915E	**Maximum number of windows already defined [RC=13]**
927E	**The virtual screen must contain at least 5 lines and 20 columns [RC=24]**
928E	**Command is not valid for virtual screen** *CMS* **[RC=12]**
1019E	**Network File System name is not allowed [RC=32]**
1020E	**Foreign host cannot be reached. The request returned return code** *rc* **and reason code** *rs* **[RC=55, 104]**
1138E	**File sharing conflict for file {***fn ft fm***|***pathname***] [RC=70]**
1214W	**File** *fn ft fm* **already locked SHARE**
1215E	**File** *fn ft fm* **is locked or in use by another user [RC=70]**
1229E	*fn ft fm* **is empty [RC=88]**
1229I	*fn ft fm* **is empty**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=31, 55, 70, 76, 99, or 100]**

1299W	**Warning: Not authorized to lock file** *fn ft fm*
1300E	**Error** *nn* **{locking|unlocking} file** *fn ft {fm|dirname}* **[RC=55, 70, 76, 99, or 100]**
2105E	**Permission is denied [24]**
2131E	**[FNAME] [FTYPE] [PNAME] {is|are} not set and NAMETYPE {CMS|BFS} is in effect [RC=24]**
2134E	**Return code** *bpxrc* **and reason code** *bpxrs* **given on call to** *rtnname* **[for path name** *pathname***] [RC=100]**
2154E	**File {***fn ft fm|pathname***} is migrated and implicit RECALL is set to OFF [RC=50]**
2155E	**DFSMS/VM error occurred during creation or recall of file {***fn ft fm|pathname***} [RC=51]**
2526E	**File or directory creation or file recall was rejected by a DFSMS/VM ACS routine; ACS routine return code** *acs rcode* **[RC = 51]**

Messages with Member Options

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

Messages with Update Options

007E	**File** *fn ft fm* **is not fixed, 80-character records [RC=32]**
007E	**File** *fn ft fm* **does not have a logical record length greater than or equal to 80 [RC=32]**
007E	**File** *fn ft fm* **does not have the same format and record length as** *fn ft fm* **[RC=32]**
007E	**File** *fn ft fm* **is not fixed record format [RC=32]**
104S	**Error** *nn* **reading file** *fn ft fm* **from disk or directory [RC=31, 32, or 55]**
174W	**Sequence error introduced in output file:** *seqno1* **to** *seqno2* **[RC=32]**
178I	**Applying** *fn ft fm*
179E	**Missing or invalid MACS card in control file** *fn ft fm*
180W	**Missing PTF file** *fn ft fm*
183E	**Invalid {CONTROL|AUX} file control card [RC=32]**
184W	**./ S not first card in update file--ignored [RC=32]**
185W	**Non numeric character in sequence field** *seqno* **[RC=32]**
186W	**Sequence number not found [RC=32]**
207W	**Invalid update file control card [RC=32]**
210W	**Input file sequence error:** *seqno1* **to** *seqno2* **[RC=32]**
317E	**Number of AUX file types in control file** *fn ft fm* **exceeds 32 [RC=32]**
570W	**Update** *ft* **specified in the UNTIL option field not found**
597E	**Unable to merge updates containing ./ S cards [RC=32]**
1262S	**Error** *nn* **opening file** *fn ft fm* **[RC=31, 55, 70, 76, 99, or 100]**

where return codes are:

0	Normal
3	LOAD must be the first subcommand
4	File is already in storage
5	Profile result is not a valid number
6	Subcommand rejected in the profile due to LOAD error, or QUIT subcommand has been issued in a macro called from the last file in the ring
12	Command is not valid for virtual screen
13	Maximum number of windows already defined
20	A character in the file name, file type, or path name is not valid

24	Invalid parameters or options
28	Source file not found (UPDATE MODE), or library not found (MEMBER option), or specified PROFILE macro does not exist, or file XEDTEMP CMSUT1 already exists
31	A rollback occurred
32	Error during updating process, or file is not a library, or library has no entries, or file is not fixed, 80 char. records or maximum number of AUX file types exceeded, or BFS file is not a regular file, or Network File System path name cannot be used
36	Corresponding minidisk or directory not accessed
50	File is DFSMS/VM migrated and automatic recall has been set to OFF (CMS SET RECALL command)
51	DFSMS/VM error
55	APPC/VM communications error or TCP/IP communications error
70	File sharing conflict or the minidisk file being opened is already open using CSL interfaces of DMSOPEN or DMSOPDBK
76	Connection error
80	An I/O error occurred while an OS data set or DOS file was being read or an OS or DOS disk was detached without being released
81	The file is an OS read-password-protected data set or a DOS file with the input security indicator on
82	The OS data set or DOS file is not BPAM, BSAM, or QSAM
83	The OS data set or DOS file has more than 16 user labels or data extents
88	File is too large and does not fit into storage, a previous MACLIB function was not finished, or an unsupported function was attempted with an empty file
99	A required system resource is not available
100	Error reading the file into storage
104	Insufficient storage available

& (Ampersand)

```
►►──&─┬────────────┬──────────────────────────────────────────────►◄
      └─subcommand─┘
```

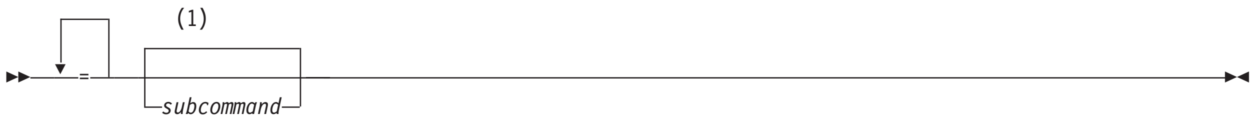
Purpose

Use an ampersand (&) at the command line before any subcommand to redisplay the subcommand.

Usage Notes

1. Using an & allows you to reexecute the subcommand just by pressing ENTER. You can also modify the command before reentering it.
2. A synonym cannot be defined for &.
3. The & subcommand must not be preceded by blanks. Also, & replaces trailing blanks with nulls in the subcommand that follows it.
4. MSGMODE must be ON for & to work properly.

= (Equal Sign)



Notes:

1 If *subcommand* is not specified, the last command, macro, or CP/CMS command is executed.

Purpose

Use the = subcommand to reexecute the last subcommand, macro, or CP/CMS command entered, or to execute a specified subcommand and *then* reexecute the last one entered.

Operands

subcommand
is any XEDIT subcommand (or any CP or CMS command, if SET IMPCMSCP ON is in effect). It is executed *before* the previous subcommand is reexecuted.

Usage Notes

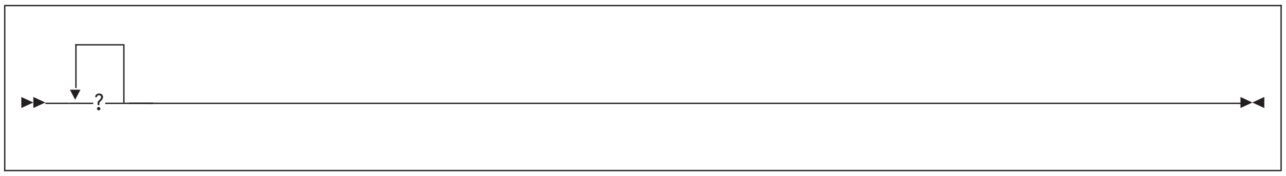
1. Multiple adjacent = subcommands (= = = =) in the command line cause the last subcommand to be executed as many times as there are equal signs.
2. The last subcommand that is being reexecuted could have been entered from the command input area on the terminal, from the console stack (through a macro), or from any of the key settings (PF, PA, or ENTER keys).
3. The editor keeps a copy of the last subcommand or macro in an *equal buffer*.
   - QUERY = displays the contents of the equal buffer.
   - The EXTRACT subcommand (EXTRACT/=/) returns the contents.
   - SET = *string* inserts *string* as the new contents of the equal buffer.
4. You can rename the = subcommand with the SET SYNONYM subcommand.
5. The editor assigns the = subcommand (with no operand) to the PF9 key. When you assign the = subcommand to a PF or PA key, the default is ONLY = (see "SET PFn" on page 308 and "SET PAn" on page 300).
6. Entering:

   ```
   = subcommand
   ```

   is useful if, for example, you enter a data line on a typewriter terminal while you are in command mode. Instead of switching to input mode and retyping the data line, you can use the following subcommand:

   ```
   = input
   ```
7. When you press ENTER and have entered nothing on the command line, the ENTER key definition, if any, is placed in the equal buffer.
8. The results of the execution of the = (Equal Sign) subcommand may not always be the same as the execution of the combination of the ? (Question Mark) subcommand and ENTER. The execution of certain defined PF keys, prefix subcommands, and so forth, can cause this difference.

? (Question Mark)



Purpose

Use the ? subcommand to display in the command area the last XEDIT subcommand (except for an = or ? subcommand), macro, or CP/CMS command executed from the command line. If none of these have been issued, the ? subcommand displays the original CMS XEDIT command.

Usage Notes

1. The subcommand that is displayed because of a ? can be reexecuted by pressing ENTER. You can also modify the command before reentering it.
2. Successive executions of ? subcommands display the previous subcommands. (The previous subcommands are maintained in a ring.)
3. A synonym cannot be defined for the ? subcommand.
4. The ? subcommand can be assigned to a PF or PA key. For example, the editor assigns the ? subcommand to the PF6 key and PA3 key (initial settings). When the ? subcommand is assigned to a PF or PA key, the default is ONLY ? (see "SET PFn" on page 308 and "SET PAn" on page 300).
5. Anything following a ? is ignored except another ?. You can specify multiple ?s to retrieve previous subcommands. For example, ??? displays the third previous subcommand.
6. Multiple subcommands ending with ? can be reexecuted from the command line by pressing ENTER. For example, DEL#NEXT#? can be used repeatedly to delete every other line from part or all of a file.
7. The results of the execution of the = (Equal Sign) subcommand may not always be the same as the execution of the combination of the ? (Question Mark) subcommand and ENTER. The execution of certain defined PF keys, prefix subcommands, and so forth, can cause this difference.
8. The ? subcommand must not be preceded by blanks. Trailing blanks in the subcommand that is displayed because of a ? subcommand are respected; they are not replaced with nulls.
9. The ? subcommand has no effect if it is issued when the CMDLINE setting is OFF or if it is issued when the console stack is not empty.

Chapter 4. Prefix Subcommands and Macros

Prefix subcommands and macros are *line* subcommands and macros you enter by typing over the five-position prefix area. You can use them to:

- Insert and delete lines
- Continually add successive lines
- Copy, move, and duplicate lines
- Extend the length of a line
- Move the current line pointer
- Display the scale on a particular line
- Display the tab settings on a particular line
- Assign a symbolic name to a line
- Shift lines left or right
- Exclude lines from display
- Redisplay (show) excluded lines

The prefix subcommands and macros are:

A	Add (equivalent to I)
C	Copy
D	Delete
E	Extend
F	Following
I	Insert
M	Move
P	Preceding
S	Show excluded lines
SCALE	Display scale
SI	Add successive lines
TABL	Display tab line
X	Exclude lines
.xxxx	Assign symbolic name
<	Shift left
/	Set current line
>	Shift right
"	Duplicate

1. To display the five-position prefix area (=====), use the subcommand:
   ```
   set prefix on right
   ```
   or
   ```
   set prefix on left
   ```
   You can use SET PREFIX NULLS to display nulls in the prefix area.
2. You can type prefix subcommands and macros over any position of the five-character prefix area.
   For example:
   ```
   ====a  (adds a line)
   a====  (adds a line)
   ==d==  (deletes a line)
   5a===  (adds 5 lines)
   3 a==  (adds 3 lines)
   ```

are valid ways to enter prefix subcommands. You can type multiple prefix subcommands and macros before pressing ENTER.

3. The prefix area is decoded in the following manner:
   - The prefix area is scanned to determine what has been entered. The scan starts at the right, looking for the first character that is different from the original prefix. If it finds one, it then scans starting at the left (if some of the prefix area has not been scanned yet), looking for the leftmost character that is different from the original prefix.
   - Any number is treated as an operand.
   - If the name starts with a letter, the name is decoded up to a non-alphabetic character.
   - If the name starts with a non-alphabetic character (a delimiter), the name is decoded up to a blank or an alphabetic character.
   - Whatever follows is treated as an operand.

   For example:

```
PREFIX     NAME       OP1     OP2     OP3

4a3 5      a           4       3       5
$XXX       $          XXX
>>5        >>          5
5a>b       a           5       >b
5>ab       >           5       ab
```

4. If line numbers are displayed in the prefix area (with SET NUMBER ON), use prefix subcommands and macros carefully. Some numbers you type in the prefix area may be ignored. Only the characters you type over *and* are different from the numbers in the prefix area are interpreted as prefix subcommands or macros.

   For example:

```
The prefix area contains           12345
You type 3a:                        123a5
The prefix subcommand is a.

The prefix area contains           12345
You type a4:                        1a445
The prefix subcommand is a4.

If you typed a44:                  1a445
or if you typed a445:              1a445
```

   the editor would still interpret the prefix subcommand as being a4, because the characters you type over must be different from the ones in the line number. For a44 to be recognized in this situation, type a44 followed by a blank.

5. With SET PREFIX NULLS and SET NUMBER ON, use prefix subcommands and macros carefully. Some numbers you type in the prefix area may be ignored.

   For example:

```
The prefix area contains               224
You type 224a:                        224a4
The prefix subcommand is a4

The prefix area contains               224
You type 224a:                        224a
The prefix subcommand is a.
```

6. When you are editing a file on multiple screens (multiple views of the same file) you can specify prefix subcommands and macros on all of the views.

   If you enter a prefix subcommand or macro for the same file line in more than one view of the file, only the prefix subcommand or macro that is lowest and furthest to the right of the virtual screen is executed.

7. If you enter a prefix subcommand or macro incorrectly, it is displayed in the prefix area, prefixed with a question mark (?). For example, if you enter XX3 (this form of the X prefix macro is not valid), the prefix area displays `?XX3`.

   If a prefix subcommand or macro causes a pending condition or if you enter an unknown prefix subcommand, the following notice is displayed in the status area:

   ```
   value pending
   ```

   where *value* is the name of the subcommand or macro you entered.

   When XEDIT processes prefix subcommands and macros, it checks for syntax errors. If it finds an error, the message is displayed and the prefix subcommand is redisplayed, preceded by a ?, in the screen on which you entered it.

   When the pending list is executed, if you entered the prefix subcommand on a line that is not valid, an error message is displayed in the screen on which the pending list was executed, and the prefix subcommand is redisplayed on the line on which you entered it, preceded by a ?.

   Prefix macros are processed when the pending list is being executed. If XEDIT finds an error in the prefix macro, the message is displayed in the screen on which the pending list is executed. If the prefix macro indicates the pending entry that was in error should be redisplayed (using SET PENDING ERROR), that entry appears in the same screen on which the error message appears.

8. The RESET subcommand (entered on the command line) cancels any pending prefix subcommands or macros, that is prefix subcommands or macros that have caused a pending notice to be displayed in the status area.

   Prefix subcommands and macros are executed before any subcommands executed by pressing a PA, PF, or ENTER key, and before any subcommands you can type in the command line, including RESET.

9. If you type a prefix subcommand to delete, copy, or move some lines and the number (*n*) you specify is greater than the number of lines left in the file, the number (*n*) is automatically adjusted to the number of lines left in the file.

10. The prefix subcommands and macros for a file are executed starting at the top of the file and moving through to the end of the file. Block commands are executed when the end of the block is reached. Prefix subcommands or macros that cannot be executed or are not recognized are left pending. In the following example, the numbers in parentheses indicate the order of processing; if you enter the following prefix subcommands:

    ```
    MM  (2)
    D   (1)
    MM  (2)
    DD  (3)
    F   (2)
    ```

    D executes first, then MM and F, followed by DD, which is left pending because it has no matching entry.

11. When you issue multiple prefix subcommands or macros, the cursor is positioned according to the one with the highest priority (see “CURSOR” on page 76).

The following lists the priorities associated with prefix subcommands and macros (and the priority assigned to changes on the screen and the ENTER key).

SI	Priority = 70
E	Priority = 60
A,I	Priority = 60
/	Priority = 50
"	Priority = 40
M	Priority = 30
C	Priority = 30
S	Priority = 30
X	Priority = 30
<,>	Priority = 30
ENTER key	Priority = 30
Screen change	Priority = 20
D	Priority = 10

For example, if you type an A and a ″ prefix subcommand and press ENTER, the cursor is positioned on the screen where the A prefix subcommand added the new line (regardless of whether the ″ preceded or followed the A on the screen).

If you type M and F prefix subcommands and a > prefix macro and press ENTER, the cursor is positioned either on the moved line or on the shifted line, depending on which was specified first on the screen.

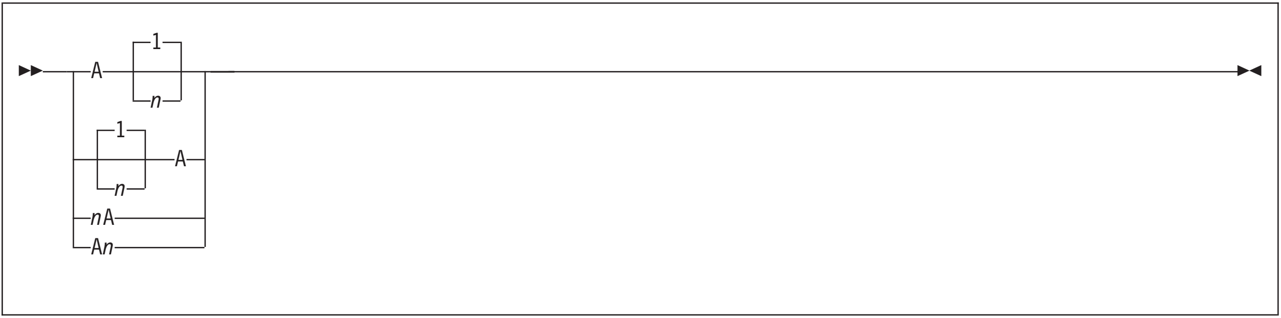
Notes for Macro Writers

1. For more information on writing prefix macros, see the commands: SET/QUERY/EXTRACT PENDING and SET/QUERY/EXTRACT PREFIX in this book and *z/VM: XEDIT User's Guide*.
2. By using the CURSOR subcommand, user-written prefix macros can specify where the cursor is to be positioned as well as a priority for this cursor movement. The cursor is positioned at the location specified that has the highest priority when all pending prefix subcommands and macros are executed.
3. You can assign synonyms for prefix macros by using the SET PREFIX subcommand with the SYNONYM option; see *z/VM: XEDIT User's Guide* for examples. SET PREFIX subcommands assigning synonyms can be entered in your PROFILE XEDIT file.

   The synonyms assigned to the XEDIT prefix macros are as follows:

Macro Synonym(s)	File Identifier
X, XX	PREFIXX XEDIT
S	PRFSHOW XEDIT
<, >, >>, <<	PRFSHIFT XEDIT
.....	SI XEDIT

A (ADD)



Purpose

Use the A prefix subcommand to add one or more lines immediately following the line in which you enter the A prefix subcommand. The A prefix subcommand is equivalent to the INSERT prefix subcommand.

Operands

A	Add one line
*n***A** **A***n*	Add *n* lines

Usage Notes

1. To add one line, enter the character A in the prefix area of the line.
2. To add more than one line, enter A*n* or *n*A in the prefix area of the *first* line of *n* lines to be added.
3. You can enter data in the newly-added lines at any time during the editing session. These lines remain in the file after it is saved or filed.
4. If you enter A in the prefix area of the End of File line, the line is added before this line.

Examples

See the “Example” section of the D prefix subcommand on page 427.

Responses

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

By default, the prefix area on each added line is highlighted. For more information, see SET COLOR PENDING.

Each added line is prefilled with the current mask. For more information, see “SET MASK” on page 287.

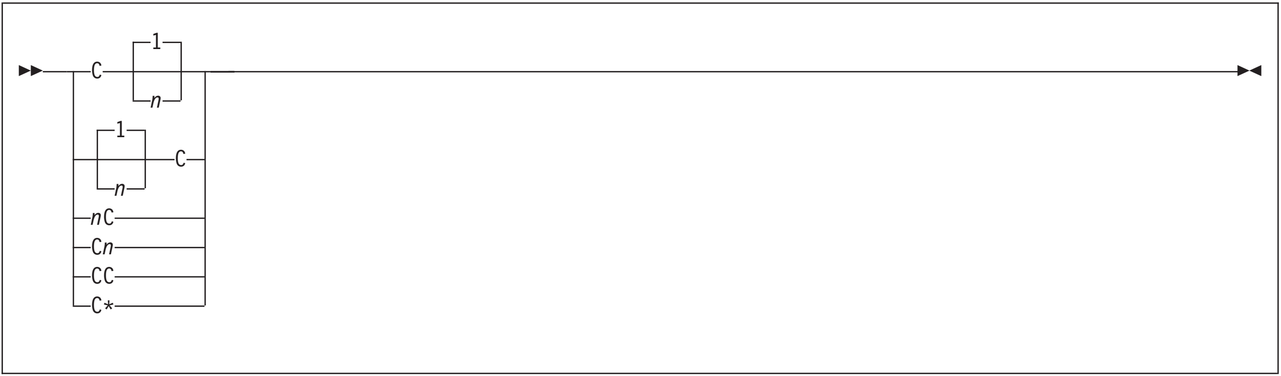
Messages and Return Codes

557S No more storage to insert lines [RC=4]
659E Invalid prefix subcommand: *prefix*

where return codes are:

4	Insufficient storage available

C (COPY)



Purpose

Use the C prefix subcommand to copy one line, a specified number of lines, or a block of lines.

Operands

C	Copy line
C*n* *n***C**	Copy *n* lines
CC	Copy block of lines
C*	Copy the rest of the lines in the file

Usage Notes

1. Whenever you enter a C prefix subcommand, enter an F or a P prefix subcommand in the prefix area of another line to indicate the destination of the copied line(s). The destination line and the line(s) to be copied can be on different screen displays but must be within the same file.
2. To copy one line, enter the character C in the prefix area of the line.
3. To copy more than one line, enter C*n* or *n*C in the prefix area of the *first* line of *n* lines to be copied.
4. To copy a block of lines, enter CC in the prefix areas of both the *first* and *last* lines, which can be on different screens but within the same file.
5. To copy the rest of the lines in a file, enter C* in the prefix area of the *first* line to be copied.

Responses

The cursor is placed on the first line that was copied (at its new location) when that line is displayed on the resulting screen. If the first copied line is not displayed on the resulting screen, the cursor is positioned on the command line.

The following notice is displayed in the status area if you have entered a C but have not yet entered an F or a P.

```
C pending...
```

The pending status lets you scroll through the file before entering the F or P.

When you enter a CC on only one line of a block, the following message is displayed in the status area:

```
CC pending...
```

This pending status lets you scroll through the file before completing the block.

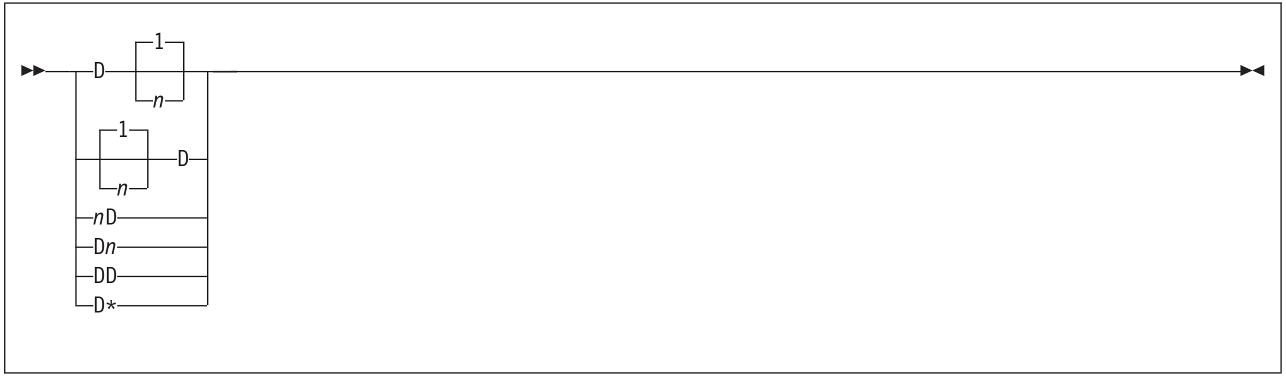
Messages and Return Codes

557S **No more storage to insert lines [RC=4]**
659E **Invalid prefix subcommand:** *prefix*
661E **Prefix** *name* **is invalid for the line on which it was entered**

where return codes are:

4	Insufficient storage available

D (DELETE)



Purpose

Use the D prefix subcommand to delete one line, a specified number of lines, or a block of lines.

Operands

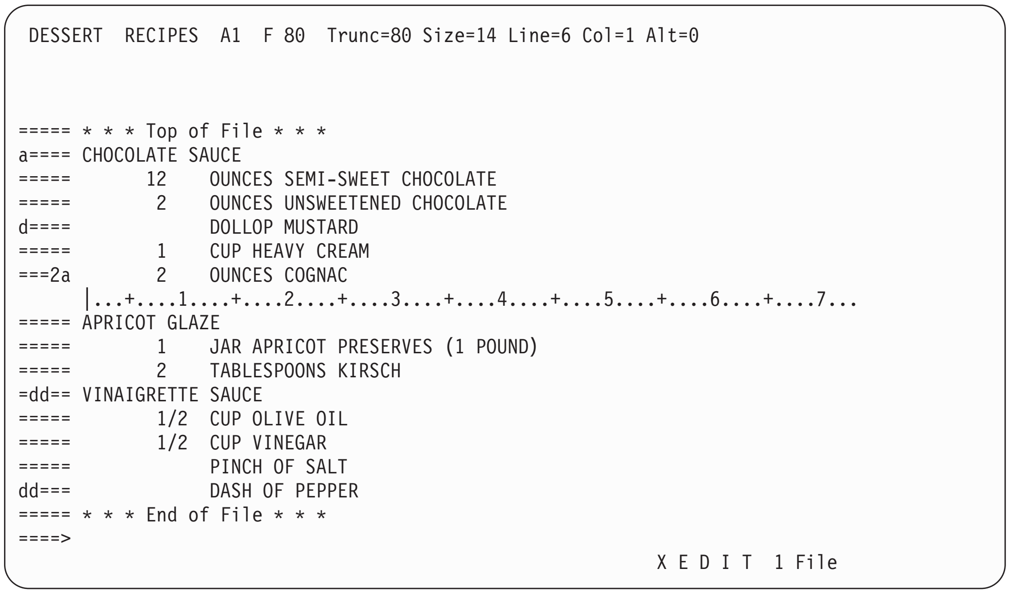
D	Delete one line
D*n*	
*n***D**	Delete *n* lines
DD	Delete block of lines
D*	Delete the rest of the lines in the file

Usage Notes

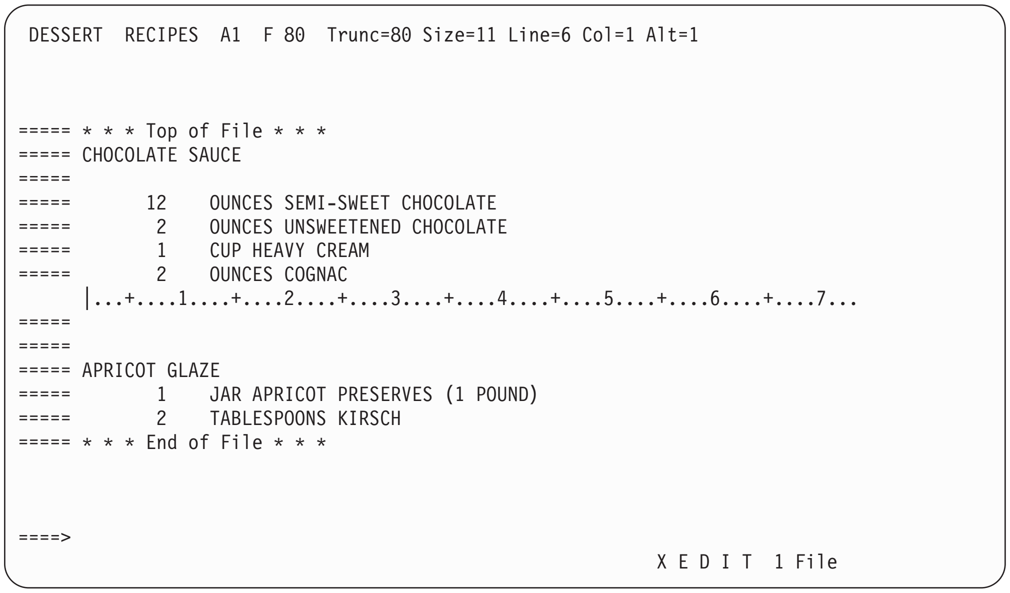
1. To delete a block of lines, enter DD in the prefix areas of both the first and last lines of the block to be deleted. The beginning and end of the block can be on different screens but must be within the same file.
2. To delete one line, enter the character D in the prefix area of the line.
3. To delete more than one line, enter D*n* or *n*D in the prefix area of the *first* line of *n* lines to be deleted.
4. To delete the rest of the lines in a file, enter D* in the prefix area of the *first* line to be deleted.

Examples

Figure 20 on page 428 is a before-and-after example of the A and D prefix subcommands. For more information, see *z/VM: XEDIT User's Guide*.



```
 DESSERT  RECIPES  A1  F 80  Trunc=80 Size=14 Line=6 Col=1 Alt=0



===== * * * Top of File * * *
a==== CHOCOLATE SAUCE
=====       12    OUNCES SEMI-SWEET CHOCOLATE
=====        2    OUNCES UNSWEETENED CHOCOLATE
d====             DOLLOP MUSTARD
=====        1    CUP HEAVY CREAM
===2a        2    OUNCES COGNAC
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== APRICOT GLAZE
=====        1    JAR APRICOT PRESERVES (1 POUND)
=====        2    TABLESPOONS KIRSCH
=dd== VINAIGRETTE SAUCE
=====        1/2  CUP OLIVE OIL
=====        1/2  CUP VINEGAR
=====             PINCH OF SALT
dd===             DASH OF PEPPER
===== * * * End of File * * *
====>
                                                            X E D I T  1 File
```



```
 DESSERT  RECIPES  A1  F 80  Trunc=80 Size=11 Line=6 Col=1 Alt=1



===== * * * Top of File * * *
===== CHOCOLATE SAUCE
=====
=====       12    OUNCES SEMI-SWEET CHOCOLATE
=====        2    OUNCES UNSWEETENED CHOCOLATE
=====        1    CUP HEAVY CREAM
=====        2    OUNCES COGNAC
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====
===== APRICOT GLAZE
=====        1    JAR APRICOT PRESERVES (1 POUND)
=====        2    TABLESPOONS KIRSCH
===== * * * End of File * * *



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

Figure 20. Prefix Subcommands A and D — Before and After

Responses

The cursor is placed on the command line. If you want the cursor to be positioned on the line following the last line deleted, you must change the priority of the ENTER (or PA or PF) key. You can do this with the CURSOR subcommand.

When you enter DD on only one line of a block of lines to be deleted, the following notice is displayed in the status area:

```
DD pending...
```

This pending status allows you to scroll through the file before completing the block.

Messages and Return Codes

659E Invalid prefix subcommand: *prefix*

661E Prefix *name* is invalid for the line on which it was entered

where return codes are:

1 Partial delete due to TOF or EOF reached during execution

E (EXTEND)

```
►►──E──────────────────────────────────────────────────────────────►◄
```

Purpose

Use the E prefix subcommand to extend a logical line by one more virtual screen line.

Usage Notes

1. After you enter an E prefix subcommand, the entire logical line, which now appears on two virtual screen lines, is treated as one line. Shifting due to character deletion and insertion affects the entire logical line.
2. The entire logical line visible on the screen does not exceed the maximum value defined in the SET VERIFY subcommand.
3. A line cannot be extended on a vertical screen (see “SET SCREEN” on page 329).

Responses

The cursor is positioned following the last nonblank character in the logical line.

Messages and Return Codes

659E Invalid prefix subcommand: *prefix*
661E Prefix *name* **is invalid for the line on which it was entered**

where return codes are:

None

F (FOLLOWING)

```
►►──F──────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the F prefix subcommand to identify the line *after* which line(s) are to be copied or moved with the C (Copy) or M (Move) prefix subcommands.

Examples

See Figure 21 on page 434 (the M prefix subcommand).

Responses

The following notice is displayed in the status area if you have entered an F prefix subcommand and not yet entered an associated C or M prefix subcommand.

```
F pending...
```

The pending status lets you scroll through the file before entering a C or M prefix subcommand.

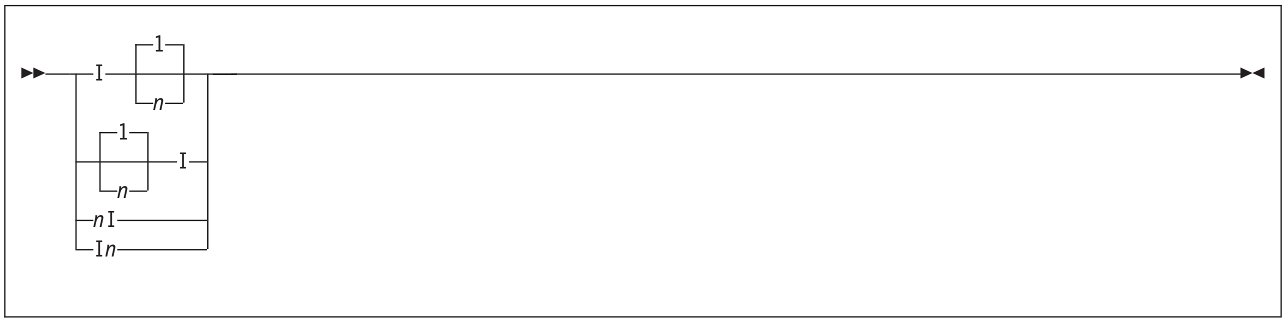
Messages and Return Codes

659E Invalid prefix subcommand: *prefix*

where return codes are:

None

I (INSERT)



Purpose

Use the I prefix subcommand to insert one or more lines immediately following the line in which you enter the I prefix subcommand.

Operands

I	Insert one line
*n***I**	
I*n*	Insert *n* lines

Usage Notes

1. The I prefix subcommand is identical to the A prefix subcommand.
2. To add one line, enter the character I in the prefix area of the line.
3. To add more than one line, enter I*n* or *n*I in the prefix area of the *first* line of *n* lines to be copied.
4. You can enter data in the newly-added lines at any time during the editing session from the prefix area. These lines remain in the file after it is saved or filed.
5. If you enter I in the prefix area of the End of File line, the line is added before this line.

Responses

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

By default, the prefix area of each line that is inserted is highlighted. For more information, see SET COLOR PENDING.

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

Messages and Return Codes

557S No more storage to insert lines [RC=4]
659E Invalid prefix subcommand: *prefix*

where return codes are:

4	Insufficient storage available

M (MOVE)

```
         ┌─1─┐
►►──┬─M──┴─n─┴─┬──────────────────────────────────────────────►◄
    │ ┌─1─┐    │
    ├─┴─n─┴─M──┤
    ├─nM───────┤
    ├─Mn───────┤
    ├─MM───────┤
    └─M*───────┘
```

Purpose

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.

Operands

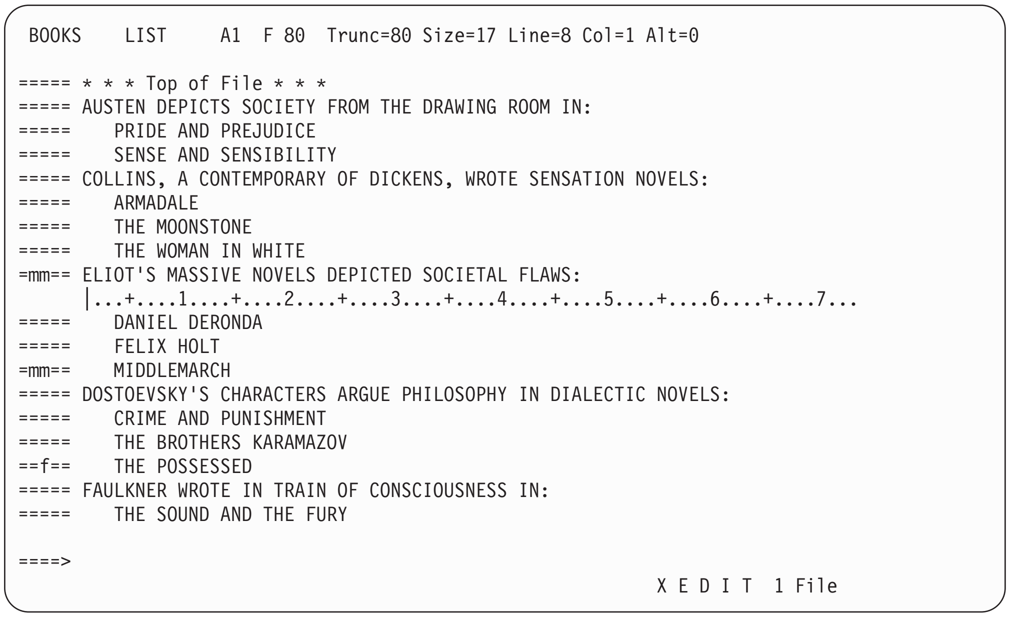
M	Move one line
M*n* *n***M**	Move *n* lines
MM	Move block of lines
M*	Move the rest of the lines in the file

Usage Notes

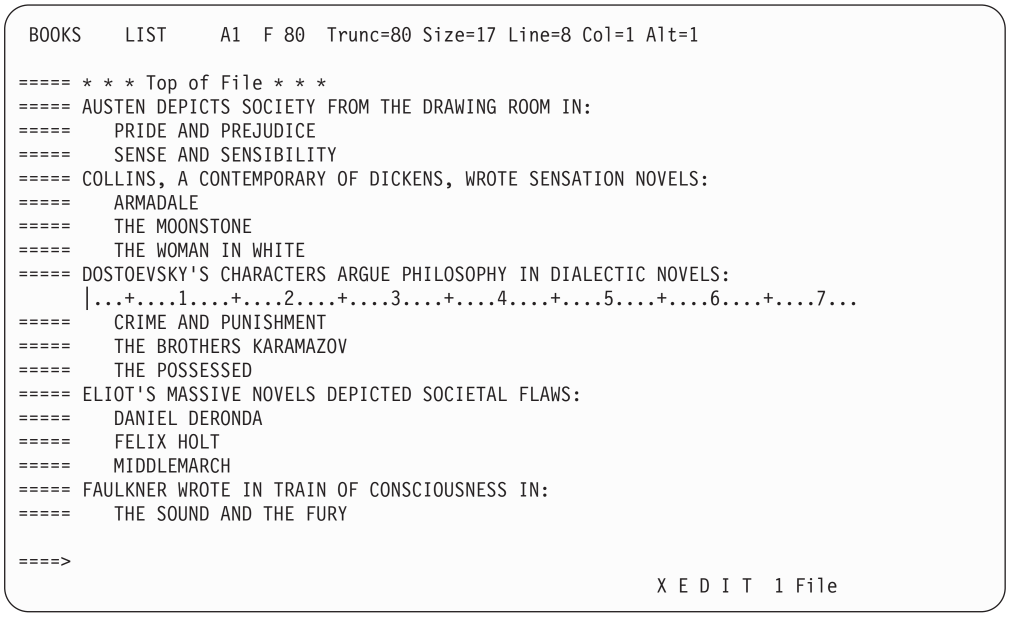
1. Whenever you enter an M prefix subcommand, enter an F or a P prefix subcommand in the prefix area of another line to indicate the destination of the lines to be moved. The destination line and the line(s) to be moved can be on different screen displays but must be within the same file.
2. To move one line, enter the character M in the prefix area of the line.
3. To move more than one line, enter M*n* or *n*M in the prefix area of the *first* line of *n* lines to be moved.
4. To move a block of lines, enter MM on both the *first* and *last* lines, which can be on different screen displays but must be within the same file.
5. To move the rest of the lines in a file, enter M* in the prefix area of the *first* line to be moved.

Examples

Figure 21 on page 434 is a before-and-after example of the M and F prefix subcommands. For more information, see *z/VM: XEDIT User's Guide*.



```
 BOOKS    LIST     A1  F 80  Trunc=80 Size=17 Line=8 Col=1 Alt=0

===== * * * Top of File * * *
===== AUSTEN DEPICTS SOCIETY FROM THE DRAWING ROOM IN:
=====    PRIDE AND PREJUDICE
=====    SENSE AND SENSIBILITY
===== COLLINS, A CONTEMPORARY OF DICKENS, WROTE SENSATION NOVELS:
=====    ARMADALE
=====    THE MOONSTONE
=====    THE WOMAN IN WHITE
=mm== ELIOT'S MASSIVE NOVELS DEPICTED SOCIETAL FLAWS:
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====    DANIEL DERONDA
=====    FELIX HOLT
=mm==    MIDDLEMARCH
===== DOSTOEVSKY'S CHARACTERS ARGUE PHILOSOPHY IN DIALECTIC NOVELS:
=====    CRIME AND PUNISHMENT
=====    THE BROTHERS KARAMAZOV
==f==    THE POSSESSED
===== FAULKNER WROTE IN TRAIN OF CONSCIOUSNESS IN:
=====    THE SOUND AND THE FURY

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



```
 BOOKS    LIST     A1  F 80  Trunc=80 Size=17 Line=8 Col=1 Alt=1

===== * * * Top of File * * *
===== AUSTEN DEPICTS SOCIETY FROM THE DRAWING ROOM IN:
=====    PRIDE AND PREJUDICE
=====    SENSE AND SENSIBILITY
===== COLLINS, A CONTEMPORARY OF DICKENS, WROTE SENSATION NOVELS:
=====    ARMADALE
=====    THE MOONSTONE
=====    THE WOMAN IN WHITE
===== DOSTOEVSKY'S CHARACTERS ARGUE PHILOSOPHY IN DIALECTIC NOVELS:
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====    CRIME AND PUNISHMENT
=====    THE BROTHERS KARAMAZOV
=====    THE POSSESSED
===== ELIOT'S MASSIVE NOVELS DEPICTED SOCIETAL FLAWS:
=====    DANIEL DERONDA
=====    FELIX HOLT
=====    MIDDLEMARCH
===== FAULKNER WROTE IN TRAIN OF CONSCIOUSNESS IN:
=====    THE SOUND AND THE FURY

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

Figure 21. Prefix Subcommands M and F — Before and After

Responses

After the move, the cursor is positioned on the first line that was moved when that line is displayed on the resulting screen. If the first line that was moved is not displayed on the resulting screen, the cursor is positioned on the command line.

The following notice is displayed in the status area if you have entered an M but have not yet entered an F or P.

```
M pending...
```

The pending status lets you scroll through the file before entering the P or F.

When you enter an MM on only one line of a block, the following notice is displayed in the status area:

```
MM pending...
```

This pending status lets you scroll through the file before completing the block.

Messages and Return Codes

557S No more storage to insert lines [RC=4]
659E Invalid prefix subcommand: *prefix*
661E Prefix *name* **is invalid for the line on which it was entered**

where return codes are:

4	Insufficient storage available

P (PRECEDING)

```
►►──P──────────────────────────────────────────────────────────►◄
```

Purpose

Use the P prefix subcommand to identify the line *before* which line(s) are to be copied or moved with the C (Copy) or M (Move) prefix subcommands.

Responses

The following notice is displayed in the status area if you have entered a P prefix subcommand and not yet entered an associated C or M prefix subcommand:

```
P pending...
```

The pending status lets you scroll through the file before entering a C or M prefix subcommand.

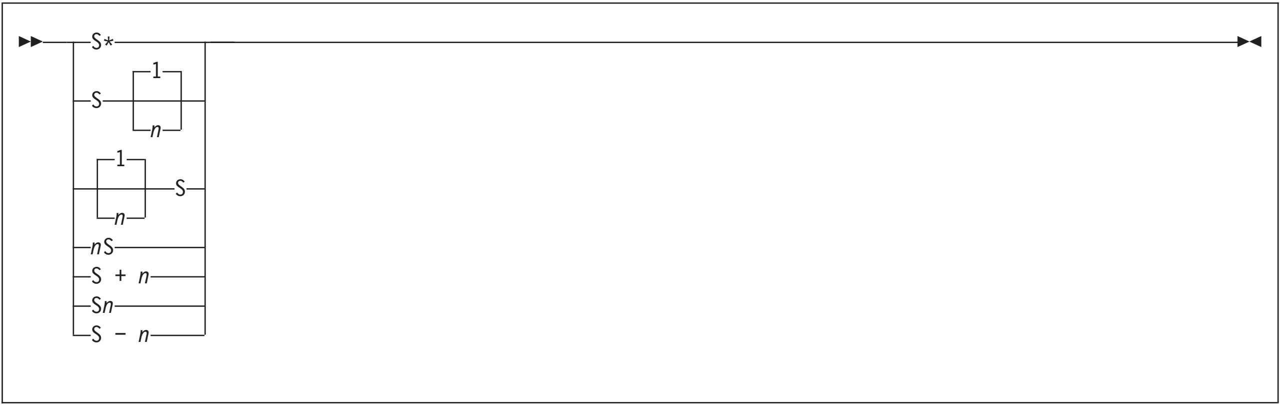
Messages and Return Codes

659E   Invalid prefix subcommand: *prefix*

where return codes are:

None

S (SHOW) Macro



Purpose

Use the S prefix macro to redisplay one or more lines you excluded with the X (Exclusive) prefix macro, the ALL macro, or other selective line editing subcommands (SET SELECT or SET DISPLAY). You can enter the S prefix macro only in the prefix area of a shadow line (see “SET SHADOW” on page 343).

Operands

S*
S Show all lines

*n***S**
S+*n*
S*n* Show the first *n* lines

S-*n* Show the last *n* lines

Usage Notes

1. When you specify *n* or +*n*, lines are redisplayed starting at the beginning of the group of excluded lines. When you specify –*n*, lines are redisplayed starting at the end of the group of excluded lines. However, they are displayed in ascending order. For example, if lines 1 through 10 are excluded, S–2 redisplays lines 9 and 10, in that order.

   If +*n* or –*n* is larger than the number of excluded lines in the group, *n* is automatically adjusted to display all of the excluded lines.
2. Redisplayed lines are included in the scope of regular editing subcommands.
3. The S prefix macro alters the selection level (see “SET SELECT” on page 333) of the redisplayed lines to the *n2* value of SET DISPLAY *n1 n2*.

Notes for Macro Writers

1. The file identifier for the S prefix macro is PRFSHOW XEDIT.

Responses

If all lines in a group of excluded lines are redisplayed, the shadow line disappears. If one or more lines remain excluded, the notice in the shadow line is adjusted accordingly. The cursor is placed on the first redisplayed line.

Messages and Return Codes

646E *Macroname* **must be invoked from the prefix area [RC=8]**
659E **Invalid prefix subcommand:** *prefix*
661E **Prefix** *name* **is invalid for the line on which it was entered**

where return codes are:

8	Subcommand must be issued from prefix area

SCALE (DISPLAY SCALE)

```
►►──SCALE──────────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the SCALE prefix subcommand to display the scale on the corresponding screen line.

Usage Notes

1. The SCALE prefix subcommand has the same effect as the subcommand:

   `set scale on n`

Responses

```
The scale looks like this:

 <...+..|.1....+....2....+....3.>..+....4T...+....5....+....6....+....7...
 .      .                       .        .
 .      .                       .        .
 .      .column pointer         .        .
 .                              .        .truncation column
 .left zone                     .right zone
```

Messages and Return Codes

659E Invalid prefix subcommand: *prefix*

where return codes are:

None

SI (STRUCTURED INPUT) Macro

```
►►──SI──────────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the SI prefix macro to add a line immediately after the line on which you specify SI and position the cursor at the column where the text on the preceding line begins. Another new line is added each time you type on the new line and press ENTER.

Usage Notes

1. SI can also be issued as a subcommand. See the SI macro.
2. Using SI, you can add a blank line in a file by making at least one change on the new line. The line is considered changed if you press the space bar or if you retype the mask characters (see "SET MASK" on page 287). Using the cursor position keys to move the cursor over a line does not change the line.
3. After SI is terminated, the cursor is positioned at the indentation column of the last added line.
4. As you add lines with SI, any data above the new line remains stationary while the data below scrolls down the screen. When the new line is one line above the bottom of the file area, adding more lines scrolls the data above the new line up the screen. Using multiple SI commands or other prefix subcommands along with SI may move the new line off the screen. The cursor may not be placed at the indentation column when you issue multiple SI subcommands.

Notes for Macro Writers

1. SI uses the console stack to adjust the current line when anything is pending on the line that will become the current line.

Examples

The following is an example of the SI macro. For more information, see *z/VM: XEDIT User's Guide*.

To add more ingredients following the line that contains the words *graham cracker crumbs*, type SI in the prefix area of that line.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=si==      1 1/2  Cups graham cracker crumbs
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

Pressing the ENTER key adds a new line. Note, the prefix area of the new line contains '. . . . .' and the cursor has been indented on the new line.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
.....      _
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

Type an ingredient on the new line and press ENTER.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
.....      8      Ounces sweetened condensed milk _
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

The '. . . . .' notice and the cursor move from the line you just typed on to another new line.

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
=====      8      Ounces sweetened condensed milk
.....      _
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

You can continue to enter ingredients by typing on the new line and pressing ENTER. When you are finished entering ingredients, press ENTER (in this example the new line was unchanged).

```
===== Chocolate-Nut Cookie Ingredients
=====
=====      1/2    Pound butter
=====      1 1/2  Cups graham cracker crumbs
=====      8      Ounces sweetened condensed milk
=====      3 1/2  Ounces coconut flakes
=====      2      Ounces chopped nuts
```

Responses

The prefix area of the new line contains '.....' and the following message is displayed in the status area:

```
   ..... pending...
```

This pending status allows you to continually add lines after you have typed on the new line.

By default, the prefix area for the line that is added is highlighted. For more information, see SET COLOR PENDING.

Each line that is added is prefilled with the current mask (see "SET MASK" on page 287).

Messages and Return Codes

529E SI is only valid in {display|editing} mode [RC=3]
659E Invalid prefix subcommand: *prefix*
661E Prefix *name* **is invalid for the line on which it was entered**

where return codes are:

3	Terminal is not a display terminal

TABL (DISPLAY TAB LINE)

```
►►──TABL────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the TABL prefix subcommand to display a T in every tab column, according to the current tab settings (see “SET TABS” on page 356), on the corresponding screen line.

Usage Notes

1. The TABL prefix subcommand has the same effect as the subcommand:

   ```
   set tabline on n
   ```
2. Use the SET TABLINE subcommand to remove the tab line.

Responses

The line displays a T in every tab column.

For example:

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

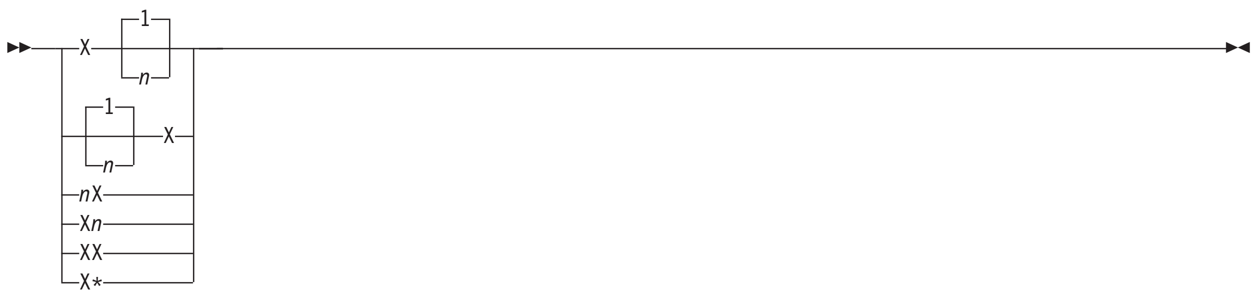
Messages and Return Codes

659E Invalid prefix subcommand: *prefix*

where return codes are:

None

X (EXCLUDE) Macro



Purpose

Use the X prefix macro to exclude from the display either one line, a specified number of lines, or a block of lines. Lines excluded from the display are also excluded from the scope of editing subcommands

Operands

X	Exclude one line from display
X*n* *n***X**	Exclude *n* lines from display
XX	Exclude a block of lines from display
X*	Exclude the rest of the lines in the file from display

Usage Notes

1. To exclude a block of lines, enter XX in the prefix area of both the first and last lines of the block, which can be on different screens but must be within the same file.
2. Use the S (SHOW) prefix macro to redisplay the excluded lines.
3. The lines excluded from display are also excluded from the scope of regular editing subcommands, unless you specify SET SCOPE ALL.
4. The X prefix macro alters the selection level of the lines excluded (see "SET SELECT" on page 333). Excluded lines are assigned a selection level that is one greater than the end of the display range. If you have entered SET DISPLAY *n* *, the X prefix macro has no effect. The X prefix macro does not alter the settings of SET SCOPE, SET SHADOW, or SET DISPLAY. For more information on selection levels, see page 455.
5. If a block of lines to be excluded includes lines *previously* excluded (that is, these lines are nested within the block), these lines remain excluded. However, their selection level remains the same. The shadow line shows the total number of lines excluded.
6. Use the X* prefix to exclude the rest of the lines in a file.

Notes for Macro Writers

1. The X prefix macro is an example of how selective line editing subcommands (SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW) can be used. The file identifier for the X prefix macro is PREFIXX XEDIT.

Responses

The cursor is placed on the first line following the excluded line(s).

When you enter XX on only one line of a block of lines to be excluded and press a key, the XX prefix macro is displayed highlighted in the prefix area, and the status area displays the following pending notice:

```
XX pending...
```

This pending status allows you to scroll through the file before completing the block.

The lines excluded are replaced with a *shadow line* (see "SET SHADOW" on page 343), which indicates the number of lines excluded. If you do not want the shadow line displayed, issue SET SHADOW OFF.

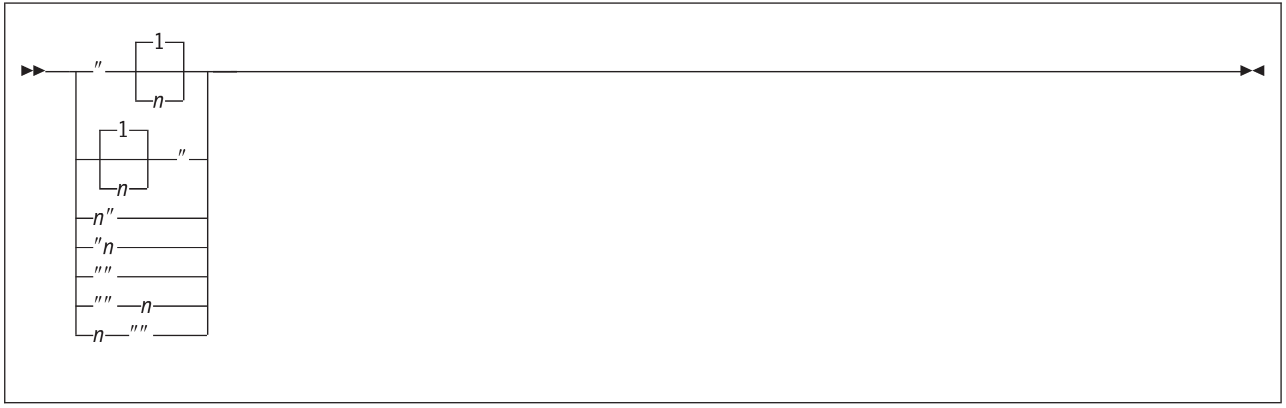
Messages and Return Codes

646E	*Macroname* **must be invoked from the prefix area [RC=8]**
659E	**Invalid prefix subcommand:** *prefix*
661E	**Prefix** *name* **is invalid for the line on which it was entered**
686E	**Synonym** *name* **not recognized by prefix macro** *macroname*

where return codes are:

8	Subcommand must be issued from prefix area

" (DUPLICATE)



Purpose

Use the " (double quote) prefix subcommand to duplicate one line or a block of lines, either one time or a specified number of times.

Operands

"
: Duplicate one line

"n
n"
: Duplicate line *n* times

""
: Duplicate block of lines

""n
n""
: Duplicate block *n* times

Usage Notes

1. To duplicate a block of lines, enter "" on both the first and last lines of the block. The beginning and end of the block can be on different screens.
2. To duplicate a block of lines *n* times, enter ""*n* or *n*"" on the first line of the block, and enter "" on the last line of the block.

Responses

The cursor is positioned on the duplicated line.

When you enter "" on only one line of a block, the following notice is displayed in the status area:

```
"" pending...
```

This pending status allows you to scroll through the file before completing the block.

Examples

For more information, see *z/VM: XEDIT User's Guide*.

Messages and Return Codes

557S No more storage to insert lines [RC=4]
659E Invalid prefix subcommand: *prefix*

"

661E Prefix *name* is invalid for the line on which it was entered

where return codes are:

4	Insufficient storage available

/ (SET CURRENT LINE)

```
►►──┬─/─┬───┬─┬──────────────────────────────────────────────────►◄
    │   └─n─┘ │
    ├─┬───┬─/─┤
    │ └─n─┘   │
    ├─/n──────┤
    └─n/──────┘
```

Purpose

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.

Operands

n is the column number in which the column pointer is to be placed.

Usage Notes

1. If you type several / prefix subcommands on the screen, the last one sets the current line.
2. If SET IMAGE ON is in effect, the cursor is placed in the first tab column of the new current line. Otherwise, it is placed in column 1.

Messages and Return Codes

659E Invalid prefix subcommand: *prefix*
661E Prefix *name* **is invalid for the line on which it was entered**

where return codes are:

None

.XXXX (SET SYMBOLIC NAME)

```
►►──.xxxx──────────────────────────────────────────────────────────────────►◄
```

Purpose

Use the *.xxxx* prefix subcommand to assign a symbolic name to a line. You can define one or more names for a line using separate *.xxxx* subcommands. You can use a symbolic name to refer to the line in subsequent target operands of XEDIT subcommands.

Operands

.*xxxx* prefix subcommand
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:
   - The *.xxxx* prefix subcommand limits the name to four characters.
   - You can delete a symbolic name for a line by using *.xxxx* to assign that name to another line.
2. The *.xxxx* prefix subcommand makes it unnecessary for you to remember or to look up the line number. You can reference a line 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 with 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 * displays all names currently defined for the file and their line numbers. The subcommand QUERY POINT displays the name(s) of the current line.

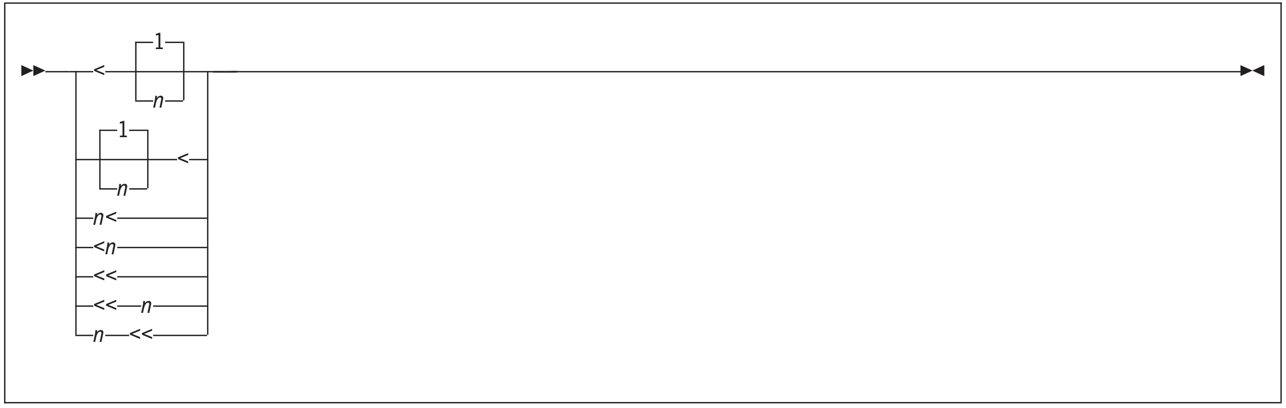
Messages and Return Codes

659E Invalid prefix subcommand: *prefix*
661E Prefix *name* **is invalid for the line on which it was entered**

where return codes are:

None

< (SHIFT LEFT) Macro



Purpose

Use the < prefix macro to shift one line or a block of lines one or more columns to the left.

Operands

<	Shift one line one column to the left
<*n* *n*<	Shift one line *n* columns to the left
<<	Shift a block of lines one column to the left
<<*n* *n*<<	Shift a block of lines *n* columns to the left

Usage Notes

1. Data shifted to the left past the zone1 column is lost. The line is padded with blanks to the right, up through the truncation column. (The < prefix macro is comparable to the SHIFT subcommand issued with the LEFT operand.)
2. To shift a block of lines one column to the left, enter << in the prefix areas of both the first and last lines of the block. The beginning and end of the block can be on different screens but must be within the same file.

   To shift a block of lines *n* columns to the left, enter <<*n* or *n*<< in either the first or last line of the block and << in the other. If you enter numbers on both the first and last lines of the block, the one closer to the end of file is used.

Notes for Macro Writers

1. The file identifier for the < prefix macro is PRFSHIFT XEDIT.

Responses

The cursor is placed on the first line shifted.

When you enter << on only one line of a block of lines to be shifted and press a key, the << prefix macro is displayed highlighted in the prefix area, and the status area displays the following pending notice:

```
<< pending...
```

This pending status lets you scroll through the file before completing the block.

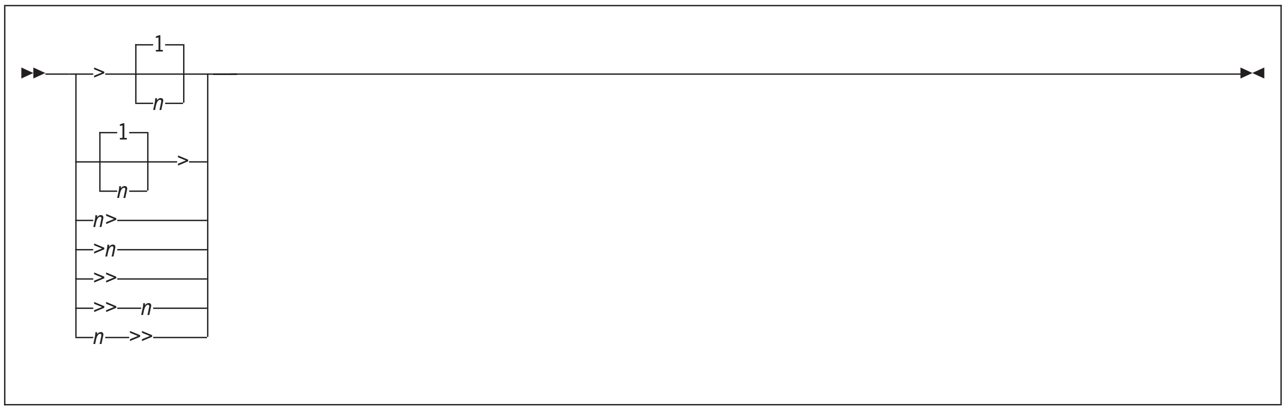
Messages and Return Codes

646E	*Macroname* **must be invoked from the prefix area [RC=8]**
659E	**Invalid prefix subcommand:** *prefix*
661E	**Prefix** *name* **is invalid for the line on which it was entered**
686E	**Synonym** *name* **not recognized by prefix macro** *macroname*

where return codes are:

8	Subcommand must be issued from prefix area

> (SHIFT RIGHT) Macro



Purpose

Use the > prefix macro to shift one line or a block of lines one or more columns to the right.

Operands

>	Shift one line one column to the right
>*n* *n*>	Shift one line *n* columns to the right
>>	Shift a block of lines one column to the right
>>*n* *n*>>	Shift a block of lines *n* columns to the right

Usage Notes

1. Shifted data that extends past the truncation column is either lost or *spilled* (see SET SPILL). The line is padded to the left with blanks. (The > prefix macro is comparable to the SHIFT subcommand issued with the RIGHT operand.)
2. To shift a block of lines one column to the right, enter >> in the prefix areas of both the first and last lines of the block. The beginning and end of the block can be on different screens but must be within the same file.

   To shift a block of lines *n* columns to the right, enter >>*n* or *n*>> in either the first or last line of the block and >> in the other. If you enter numbers on both the first and last lines of the block, the one closer to the end of file is used.

Notes for Macro Writers

1. The file identifier for the > prefix macro is PRFSHIFT XEDIT.

Responses

The cursor is placed on the first line shifted.

When you enter >> on only one line of a block of lines to be shifted and press a key, the >> prefix macro is displayed highlighted in the prefix area, and the status area displays the following pending notice:

```
>> pending...
```

>

This pending status lets you scroll through the file before completing the block.

Messages and Return Codes

646E	*Macroname* **must be invoked from the prefix area [RC=8]**
659E	**Invalid prefix subcommand:** *prefix*
661E	**Prefix** *name* **is invalid for the line on which it was entered**
686E	**Synonym** *name* **not recognized by prefix macro** *macroname*

where return codes are:

8	Subcommand must be issued from prefix area

Appendix A. File Type Defaults

The following figures show the special file types the editor recognizes and indicates the default settings the editor supplies for logical record length, logical tabs, truncation, and so on.

FILE TYPE	SERIAL	TRUNC	LRECL	RECFM	VERIFY	ESCAPE	CASE	SPILL	IMAGE
$EXEC	ON	72	80	F	72	/	M	OFF	ON
$REXX	ON	72	80	F	72	/	M	OFF	ON
$XEDIT	ON	72	80	F	72	/	M	OFF	ON
AMSERV	ON	72	80	F	T	/	U	OFF	ON
ASM3705	ON	71	80	F	T	/	U	OFF	ON
ASSEMBLE	ON	71	80	F	72	/	U	OFF	ON
BASDATA	OFF	255	255	V	T	/	M	OFF	ON
BASIC	OFF	156	156	V	T	/	M	OFF	ON
CNTRL	OFF	80	80	F	T	/	U	OFF	ON
COBOL	ON	72	80	F	T	/	U	OFF	ON
COPY	ON	71	80	F	T	/	U	OFF	ON
DLCS	ON	72	80	F	72	/	U	OFF	ON
DIRECT	ON	72	80	F	T	/	U	OFF	ON
ESERV	ON	71	80	F	T	/	U	OFF	ON
EXEC	OFF	130	130	V	T	/	U	OFF	ON
FORTRAN	ON	72	80	F	T	/	U	OFF	ON
FREEFORT	OFF	81	81	V	T	/	U	OFF	ON
GCS	OFF	130	130	V	T	/	U	OFF	ON
GROUP	ON	71	80	F	72	/	U	OFF	ON
JOB	OFF	80	80	F	T	+	U	OFF	ON
LISTING	OFF	121	121	V	T	/	U	OFF	ON
MACLIB	OFF	71	80	F	72	/	U	OFF	OFF
MACRO	ON	71	80	F	72	/	U	OFF	ON
MEMBER	ON	71	80	F	72	/	U	OFF	ON
MEMO	OFF	80	80	V	T	/	M	WORD	ON
MODULE	OFF	80	80	V	72	/	M	OFF	OFF
NAMES	OFF	255	255	V	T	/	M	OFF	ON
NETLOG	OFF	255	255	V	T	/	M	OFF	ON
NOTE	OFF	132	132	V	T	/	M	WORD	ON
NOTEBOOK	OFF	132	132	V	T	/	M	WORD	ON
PASCAL	OFF	72	72	V	T	/	M	OFF	ON
PLI	ON	72	80	F	T	/	U	OFF	ON
PLIOPT	ON	72	80	F	T	/	U	OFF	ON
REXX	OFF	255	255	V	T	/	U	OFF	ON
SCRIPT	OFF	132	132	V	T	/	M	WORD	CANON
TEXT	OFF	80	80	F	72	/	M	OFF	OFF
UPDATE	ON	71	80	F	72	/	U	OFF	ON
UPDT	ON	71	80	F	72	/	U	OFF	ON
VSBASIC	OFF	80	80	F	T	/	U	OFF	ON
VSBDATA	OFF	132	132	V	T	/	U	OFF	ON
XEDIT	OFF	255	255	V	T	/	U	OFF	ON
Other	OFF	80	80	F	**	/	U	OFF	ON

Figure 22. CMS Record File System File Type Defaults

VERIFY = T
: means verify is trunc column

VERIFY = **
: means verify is screen size (or lrecl if screen size is less than lrecl).

```
BFSLINE   SERIAL TRUNC  LRECL  RECFM  VERIFY  ESCAPE  CASE  SPILL  IMAGE

lrecl     OFF    lrecl  lrecl  F      **       /       M     OFF    OFF
other     OFF    W      W      V      **       /       M     OFF    OFF

TRUNC, LRECL = W means they are equal to WIDTH.
```

Figure 23. Byte File System (BFS) File Type Defaults

VERIFY = **
means verify is screen size (or lrecl if screen size is less than lrecl).

Tab settings for BFS files default to ‘other’.

FILE TYPE	TAB SETTINGS
$EXEC	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
$REXX	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
$XEDIT	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
AMSERV	2 5 10 15 20 25 30 35 40 45 50 55 60
ASM3705	1 10 16 30 35 40 45 50 55 60 65 70
ASSEMBLE	1 10 16 30 35 40 45 50 55 60 65 70
BASDATA	1 7 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
BASIC	1 7 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
CNTRL	1 5 8 17 27 31
COBOL	1 8 12 20 28 36 44 68 72 80
COPY	1 10 16 30 35 40 45 50 55 60 65 70
DLCS	1 4 7 10 13 16 19 22 25 31 37 43 49 55
DIRECT	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
ESERV	2 5 10 15 20 25 30 35 40 45 50 55 60
EXEC	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
FORTRAN	1 7 10 15 20 25 30 80
FREEFORT	9 15 18 23 28 33 38 81
GCS	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
GROUP	1 10 16 30 35 40 45 50 55 60 65 70
JOB	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
LISTING	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
MACLIB	1 10 16 30 35 40 45 50 55 60 65 70
MACRO	1 10 16 30 35 40 45 50 55 60 65 70
MEMBER	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
MEMO	1 10 16 30 35 40 45 50 55 60 65 70
MODULE	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NAMES	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NETLOG	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NOTE	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
NOTEBOOK	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
PASCAL	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
PLI	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
PLIOPT	2 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
REXX	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
SCRIPT	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
TEXT	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
UPDATE	1 10 16 30 35 40 45 50 55 60 65 70
UPDT	1 10 16 30 35 40 45 50 55 60 65 70
VSBASIC	7 10 15 20 25 30 80
VSBDATA	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120
XEDIT	1 4 7 10 13 16 19 22 25 31 37 43 49 55 79 80
Other	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120

Appendix B. Effects of Selective Line Editing Subcommands

This appendix addresses those *automatic* sets and differences in subcommand operation when invoked from different selective line editing environments.

You can use selective line editing in a macro to control both the action of the editor and the screen display. It consists of four subcommands: SET SELECT, SET DISPLAY, SET SCOPE, and SET SHADOW, which work together as follows. SET SELECT assigns a *selection level* or value to one or more lines in a file. You can logically group lines by assigning them the same selection level. SET DISPLAY with SET SELECT displays those lines having the same selection level. SET SCOPE defines the set of lines the editor can act on. SCOPE DISPLAY is the initial setting and, as such, restricts editor action to only those lines SET DISPLAY defines. By default, SET SHADOW displays a notice indicating how many lines are not being displayed in the physical position of the excluded lines in the file. If SET SHADOW is OFF, only those lines defined by SET DISPLAY appear on the screen, and no shadow lines indicate where lines are not being displayed. Some subcommands automatically cause specific *SETs* to be made within the file when they are invoked from a selective line editing environment.

AUTOMATIC SELECTION LEVEL ASSIGNMENT:

The initial selection level for all lines in a file is 0. When you add new lines to a file, they are automatically assigned a selection level. That selection level depends on the subcommand that added the new lines. A list of those subcommands and how each affects selection level assignment follows.

`ADD`	New line has selection level of *n1* (the first selection level defined for inclusion in the screen display) in SET DISPLAY *n1 n2*. (Also applies to A and I prefix subcommands.)
`COPY`	Copied lines have the same selection level(s) as they do in their original position in the file. (Also applies to C prefix subcommand.)
`DUPLICAT`	Duplicated lines have the same selection level(s) as they had before duplication. (Also applies to prefix subcommand.)
`GET`	Lines inserted in the file with the GET subcommand have selection level *n1* (the first selection level defined for inclusion in the screen display) in SET DISPLAY *n1 n2*.
`INPUT`	New line has selection level of *n1* (the first selection level defined for inclusion in the screen display) in SET DISPLAY *n1 n2*.
`JOIN`	Joined lines have the same selection level as the original set of lines with which they are joined.
`MERGE`	Merged lines have the same selection level as the lines with which they are merged.
`MOVE`	Moved lines retain their original selection level(s). (Also applies to M prefix subcommand.)
`RECOVER`	Recovered lines have a selection level of *n1* (the first selection level defined for inclusion in the screen display) in SET DISPLAY *n1 n2*.
`REPLACE`	When you enter the REPLACE subcommand with text, the new line has the same selection level as the line it replaces. Entering the REPLACE subcommand without text deletes the current line and causes you to enter input mode. The selection level of the new lines inserted while you are in input mode is *n1* in SET DISPLAY *n1 n2*.
`SET SPILL`	New line(s) created because of being spilled has selection level *n1* (the first selection level defined for inclusion in the screen display) in SET DISPLAY *n1 n2*.

`SPLIT`	New line(s) created by a split has the same selection level(s) as the original line.

HOW THE SCOPE SETTING AFFECTS THE ACTION OF SOME SUBCOMMANDS:

The action of some subcommands also depends on the SCOPE setting. These subcommands can be divided into two functional groups. Those that perform:

- Target processing or searches and
- Operations on file lines.

Target Processing and Searches:

Target searches are performed only on lines within the current scope. If SCOPE DISPLAY is in effect, the target search looks at only displayed lines. This is true for all types of targets: an absolute line number, a relative displacement from the current line, a line name, a simple string expression, or a complex string expression. Line movement using targets is done as if lines outside the scope have been removed from the file. For example, NEXT or +1 may go from line 20 to line 40 if lines 21 to 39 are outside the display range. If SCOPE ALL is in effect, the editor acts on the entire file.

Table 2 through Table 5 on page 457 shows the effect of the scope setting on some subcommands.

Table 2. Effects of Scope Setting on Some Subcommands

Subcommand	Scope Display	Scope All
`BOTTOM`	Last line of the scope becomes the current line. (Line must be displayed.)	Last line of the file becomes the current line. (Line may or may not be displayed.)
`CLOCATE /string/`	String must be in the scope to be located. (Line containing string must be displayed.)	String must be in the file to be located. (Line containing string may or may not be displayed.)
`DOWN, NEXT`	Next line in the scope becomes the current line. (Line must be displayed.)	Next line in the file becomes the current line. (Line may or may not be displayed.)
`FIND or FINDUP`	Searches forward or backward in the file for the first line within the scope that starts with the text specified in the operand. (Line must be displayed.)	Searches forward or backward in the file for the first line that starts with the text specified in the operand. (Line may or may not be displayed.)
`LOCATE +1`	Next line in the scope is located. (Line must be displayed.)	Next line in the file is located. (Line may or may not be displayed.)
`NFIND or NFINDUP`	Searches forward or backward in the file for the first line within the scope that does not start with the text specified in the operand. (Line must be displayed.)	Searches forward or backward in the file for the first line that does not start with the text specified in the operand. (Line may or may not be displayed.)
`UP`	Previous line in the scope becomes the current line. (Line must be displayed.)	Previous line in the file becomes the current line. (Line may or may not be displayed.)

This concept applies for every subcommand taking a target except SET RANGE, SORT, and the macro ALL. These are special cases that operate outside the SCOPE; they execute as if SCOPE ALL were in effect, regardless of the SCOPE setting. Table 3 on page 457, shows some examples.

Table 3. Effects of Scope Setting on Some Subcommands

Subcommand	Scope Display	Scope All
`SET RANGE :1 :20`	RANGE is set from line 1 to 20 whether 1 and 20 are in the scope or not. (Lines 1 and 20 may not be displayed.)	Same as SCOPE DISPLAY
`SORT /NY/ 1 5`	Columns 1 to 5 are sorted into ascending sequence for all lines in the file from the current line up to but not including the line containing the string NY. (Lines may or may not be displayed.)	Same as SCOPE DISPLAY
`ALL /SET/`	All lines in the file with string SET are selected for editing (Lines may or may not be displayed.)	Same as SCOPE DISPLAY

Operations:

The SCOPE setting affects the operation of many subcommands. See Table 4 for a few examples.

Table 4. Effects of Scope Setting on Some Subcommands

Subcommand	Scope Display	Scope All
`CHANGE/Record/Line/*`	*Record* is changed to *Line* in all lines in the scope from the current line until the end of the file. (Lines must be displayed.)	*Record* is changed to *Line* in all lines in the file from the current line until the end of the file. (Lines may or may not be displayed.)
`MOVE 3 /DATA/`	Three lines in the scope starting with the current line are moved after the line containing the string DATA. (The three lines must be displayed.)	Three lines in the file starting with the current line are moved after the line containing the string DATA. (The three lines may or may not be displayed.)
`UPPERCAS 20`	20 lines in the scope starting from the current line are translated to uppercase. (Lines must be displayed.)	20 lines in the file starting from the current line are translated to uppercase. (Lines may or may not be displayed.)

Except for the SORT subcommand and the ALL macro (see previous description), this concept applies to every subcommand performing an operation, including prefix subcommands and macros.

ISSUING PREFIX SUBCOMMANDS AND MACROS FROM A SHADOW LINE:

If SCOPE DISPLAY is in effect (the default), all prefix subcommands and macros you enter on a shadow line are invalid with the exception of the ones in Table 5.

Table 5. Effects of Scope Setting on Some Subcommands

Subcommand	Scope Display	Scope All
`A`	Line is added after the last line the shadow line represents.	Line is added after the first line the shadow line represents.
`F`	Line(s) is moved or copied after the last line the shadow line represents.	Line(s) is moved or copied after the first line the shadow line represents.
`I`	Line is inserted after the last line the shadow line represents.	Line is inserted after the first line the shadow line represents.
`P`	Line(s) is moved or copied before the first line the shadow line represents.	Line(s) is moved or copied before the first line the shadow line represents.

Table 5. Effects of Scope Setting on Some Subcommands (continued)

Subcommand	Scope Display	Scope All
S	Line(s) is redisplayed.	Line(s) is redisplayed.

If SCOPE ALL is in effect, you can enter all prefix subcommands and macros except SI on a shadow line. The operation requested is performed on file lines whether they are displayed or not. For example, entering D3 on a shadow line deletes the next three lines in the file whether these lines are represented by a shadow line or are displayed.

For prefix subcommands and macros with block operations, how shadow lines (excluded lines) within the block are handled depends on the SCOPE setting. See the example below.

Example:

In the following segment of a file, a shadow line falls within a block prefix subcommand entry.

```
===== The resort is comprised of 500 acres.
===== It has a private pond, waterfalls, brooks, and woodlands.
===== It is just one step away from hiking, swimming, and skating.
=dd== And it is being offered
===== --------------------  1  line(s) not displayed  --------------------
=dd== For more information, please call.
```

With SCOPE DISPLAY, the execution of the block entry does not affect the shadow line. Note, the shadow line remains in the file in this instance.

```
===== The resort is comprised of 500 acres.
===== It has a private pond, waterfalls, brooks, and woodlands.
===== It is just one step away from hiking, swimming, and skating.
===== --------------------  1  line(s) not displayed  --------------------
```

However, when the same block entry is executed with SCOPE ALL in effect, the execution of the block entry affects the shadow line.

```
===== The resort is comprised of 500 acres.
===== It has a private pond, waterfalls, brooks, and woodlands.
===== It is just one step away from hiking, swimming, and skating.
===== * * * End of File * * *
```

Appendix C. CMS Editor (EDIT) Migration Mode

To edit a file in EDIT migration mode, issue the CMS command EDIT. The XEDIT editor automatically places you in EDIT migration mode, which you can issue all the EDIT subcommands; you can also issue any XEDIT subcommand that does not have the same name as an EDIT subcommand.

In addition, EDIT migration mode provides full-screen editing capabilities, that is, you can type over data on any file line displayed on the screen.

Usage Notes

1. The following EDIT subcommands are executed in EDIT migration mode the same way they are executed under the CMS editor:

   ```
   ALTER       FNAME       PROMPT      STACK
   AUTOSAVE    FORMAT      QUIT        TABSET
   BACKWARD    FORWARD     RECFM       TOP
   BOTTOM      GETFILE     RENUM       TRUNC
   CASE        IMAGE       REPEAT      TYPE
   CHANGE      INPUT       REPLACE     UP
   CMS         LINEMODE    RESTORE     X,Y
   DELETE      LOCATE      RETURN      ZONE
   DOWN        LONG        REUSE       ?
   DSTRING     NEXT        SAVE        $XXXX *
   FIND        OVERLAY     SERIAL
   FMODE       PRESERVE    SHORT.
   ```

   * When the editor parses input that begins with *$*, it interprets the input as an EXEC and ignores any IMPCMSCP or MACRO settings.
2. The following EDIT subcommands are executed slightly differently in EDIT migration mode:

   `SCROLL/SCROLLUP`

   Under the CMS editor, these subcommands scroll the file one full screen. In EDIT migration mode, they scroll the screen one full screen minus one line, so the last (or first) line on the previous screen appears on the new display.

   `VERIFY`

   Under the CMS editor, the operands ON and OFF are ignored if the VERIFY subcommand is issued from a display terminal. In EDIT migration mode, ON and OFF are handled the same way on a display as they are on a typewriter terminal.
3. You can issue any XEDIT subcommand that does not appear in the preceding list in EDIT migration mode.
4. In EDIT migration mode, you can issue the XEDIT subcommand HELP to request information on EDIT subcommands only. You cannot request a HELP display for XEDIT subcommands.
5. In the EDIT command, the default file mode is * instead of A1.
6. The screen differs from the CMS editor's screen in the following ways:
   a. The file identification line (the first line of the screen) contains additional information: the truncation column (Trunc=nn); the current number of lines in the file (Size=nn); the file line number of the current line (Line=nn); the column number of the current column (Col=nn), and the alteration count (Alt=nn).
   b. The command line contains an arrow (====>).

   c. The lower right hand corner displays the status of the editing session, for example, input mode or edit mode.

7. When you issue EDIT, the line-end character default is ON. When editing files that contain characters the editor recognizes as line-end characters, you may want to SET LINEND OFF or change it to another character. See SET LINEND. In line mode, you must issue a CP TERMINAL LINEND OFF to cancel the effect of the default line-end character. For more information on the TERMINAL command, see *z/VM: CP Commands and Utilities Reference*.

Notes for Macro Writers

The old CMS editor is no longer supported. When an EDIT command is issued from an EXEC file specifying the OLD option, an error message will be displayed. If the OLD option is not specified, XEDIT in EDIT migration mode is invoked.

To invoke EDIT migration mode, you must issue the following statement in your EXEC file:

```
EXEC EDIT ...
```

Appendix D. Migrating from EDIT to XEDIT

Table 6 lists the EDIT subcommands and their XEDIT counterparts. Many of the subcommand names are the same; however, the operands are usually different. See the subcommand descriptions in this book for complete information on using the XEDIT subcommands.

Table 6. EDIT Migration Chart

If you used this EDIT command:	**Now use this XEDIT subcommand:**
ALTER	ALTER
AUTOSAVE	SET AUTOSAVE
BACKWARD	UP
BOTTOM	BOTTOM
CASE	SET CASE
CHANGE	CHANGE (G not supported)
CMS	CMS
DELETE	DELETE
DOWN	DOWN
DSTRING	DELETE
FILE	FILE
FMODE	SET FMODE
FNAME	SET FNAME
FORMAT	SET TERMINAL
FORWARD	DOWN
GETFILE	GET
IMAGE	SET IMAGE
INPUT	INPUT
LINEMODE	Not supported
LOCATE	LOCATE
LONG	SET MSGMODE ON LONG
NEXT	NEXT
OVERLAY	OVERLAY
PRESERVE	PRESERVE
PROMPT	Not supported
QUIT	QUIT
RECFM	SET RECFM
RENUM	RENUM
REPEAT	REPEAT (repeat any previous subcommands)
REPLACE	REPLACE
RESTORE	RESTORE
RETURN	RETURN
REUSE (=)	=
SAVE	SAVE
SCROLL	FORWARD
SCROLLUP	BACKWARD
SERIAL	SET SERIAL
SHORT	SET MSGMODE ON SHORT
STACK	STACK (STACK string not supported)
TABSET	SET TABS
TOP	TOP
TRUNC	SET TRUNC
TYPE	TYPE (second operand not supported)
UP	UP
VERIFY	SET VERIFY

Table 6. EDIT Migration Chart (continued)

If you used this EDIT command:	Now use this XEDIT subcommand:
`X or Y`	`Can be done through SET SYNONYM and REPEAT`
`ZONE`	`SET  ZONE`
`?`	`?`
`nnnn`	`:nnnn  (nnnn is equivalent to +nnnn)`
`$DUP`	`DUPLICATE`
`$MOVE`	`MOVE`

Appendix E. Optimizing Macros

The XEDIT macro VMFOPT can improve the performance of XEDIT macros. On the other hand, the VMFDEOPT macro can restore a macro optimized by VMFOPT to its original form.

The following XEDIT macros have already been optimized:

```
CMSEDIT
VMFDEOPT
VMFOPT
```

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 (with VMFDEOPT) and then reoptimized (with VMFOPT).

Note: VMFOPT and VMFDEOPT satisfactorily execute only on the macros listed above.

```
*%OPTIMIZED AT 14:13:12 ON 80/01/29
*%       N O T I C E:
*% THIS MACRO HAS BEEN OPTIMIZED USING THE XEDIT MACRO - VMFOPT
*% DE-OPTIMIZE THIS MACRO BEFORE MAKING ANY CHANGES USING - VMFDEOPT
```

VMFDEOPT Macro

```
►►──VMFDEOPT──────────────────────────────────────────────────────────────────►◄
```

Purpose

The VMFDEOPT macro restores to their original form macros VMFOPT optimized.

After the VMFDEOPT macro is executed, all &GOTO statements are restored to their original form, and any extra lines VMFOPT added (for example, the optimization statement) are removed.

When you wish to change an optimized macro, first deoptimize it with VMFDEOPT and then, after the changes are made, reoptimize it with VMFOPT.

VMFOPT Macro

```
►►──VMFOPT────────────────────────────────────────────────────────────────────►◄
```

Purpose

The VMFOPT macro improves performance by:

- Replacing labels with line numbers in EXEC 2 &GOTO statements. In other words, an EXEC 2 statement &GOTO –LABEL is replaced with 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 the label name (EXIT) is kept, so the macro can be deoptimized, if necessary.
- Recomputing existing &GOTO linenumber statements whose targets may have shifted because of extra lines VMFOPT inserted.
- Optimizing label variables of the form &GOTO –&X. Targets for label variables can be declared as label constants, using the optimizer control command *%LABELS. This command causes VMFOPT to insert EXEC 2 statements that set up special variables. The name of these special variables contains the unprintable character X'E0' to avoid confusion with user-defined variables.

For example:

```
Prior to optimizing:

*%LABELS -BOTTOM -CANCEL -CASE -CHANGE -CLINE -CMD

When optimized:

*%LABELS -BOTTOM -CANCEL -CASE -CHANGE -CLINE -CMD  SYN -CMS
 &STACK LIFO  187 194 199 221 265 287
 &READ VARS  & -BOTTOM & -CANCEL & -CASE & -CHANGE & -CLINE & -CMD
```

Appendix F. Using Double-Byte Character Sets

Many languages have more characters than can be displayed using one-byte codes (KANJI, for example). A Double-Byte Character Set (DBCS) represents these languages. You can display these characters on terminals that support Double-Byte Character Sets, such as the IBM 5550 Multistation. Each double-byte character occupies two columns on the screen. DBCS characters and characters from languages with one-byte codes can be mixed within a string.

Specifying DBCS Characters in a File

In XEDIT, to distinguish DBCS characters from one-byte EBCDIC characters, DBCS strings are enclosed with a shift-out (SO) character and a shift-in (SI) character.

Character	Hexadecimal	Description
␎	X'0E'	Shift-out (SO)
␏	X'0F'	Shift-in (SI)

Key to Conventions Used in This Section

Character	Represents a
e	Lowercase EBCDIC character
E	Uppercase EBCDIC character
Ｅ	One DBCS character (occupies two columns)
ＡＢＣ	A DBCS string

Editing DBCS Strings

Purpose

For you to see DBCS strings and for XEDIT to recognize DBCS strings in a file, the ETMODE setting must be ON.

The initial setting is based on whether the terminal can display double-byte characters. If it can, the initial setting is ON; if not, the setting is OFF.

In order for XEDIT to properly display and manipulate DBCS strings, when ETMODE is ON, it does the following:

- Drops contiguous shift-out and shift-in characters after executing a subcommand (with a few exceptions)
- Generates proper pairings of shift-out and shift-in characters after executing a subcommand
- Provides special consideration for shift-out and shift-in when doing a target search

Dropping Contiguous Shift-Out and Shift-In Characters

When you issue a subcommand, XEDIT checks for contiguous shift-out and shift-in characters the operation generates. Contiguous shift-out and shift-in characters are deleted and the remainder of the line shifts to the left two bytes (except with the CREPLACE subcommand, which changes these characters to blanks). For example, if you join lines 10 and 11 at column 12:

```
      |...+....1....+....2....+....3....+....4....
00010 eee␎ＡＡＡ␏
00011 ␎ＢＢＢ␏eee
```

The result is:

```
      |...+....1....+....2....+....3....+....4....
00010 eee␎ＡＡＡＢＢＢ␏eee
```

The contiguous shift-out and shift-in characters have been removed.

Pairing Shift-Out and Shift-In Characters

XEDIT properly pairs shift-out and shift-in characters after executing a subcommand. If a subcommand results in the deletion of a shift-out or shift-in character, the appropriate character is inserted so a DBCS string is always between shift-out and shift-in characters. For example, if you split lines 10 and 11 at column 11:

```
      |...+....1....+....2....+....3....+....4....
00010 eee␎ＡＡＡＢＢＢ␏eee
```

The result is:

```
      |...+....1....+....2....+....3....+....4....
00010 eee␎ＡＡＡ␏
00011 ␎ＢＢＢ␏eee
```

Target Searches with Shift-Out and Shift-In Characters

When XEDIT scans a file for a string target, an SO or SI is ignored in the following cases:

- The first character of a string target is an SO.
- The last character of a string target is an SI.
- The SO or SI is specified immediately preceding or immediately following:
  - An arbitrary character when ARBCHAR is ON
  - An extended arbitrary character when ETMODE and ETARBCH are ON

For example:

`LOCATE /␎Ａ␏/` locates ␎ＢＡＢ␏.

The shift-out and shift-in characters of the target are ignored during the search.

Using XEDIT Subcommands with DBCS

The following subcommands can be used in extended mode (ETMODE ON).

Add	PREServe	SET MSGLine	STATus
ALL	PURge	SET MSGMode	SUPerset
ALter	PUT, PUTD	SET NAMetype	TOP
BAckward	Query	SET NONDisp	TRAnsfer
Bottom	QUIT	SET NULls	Type
CANCEL	READ	SET NUMber	Up
CAppend	RECover	SET PA*n* *.XXXX*	UPPercas
CDelete	REFRESH	SET PACK	Xedit
CFirst	RENum	SET PENDing	&
Change	REPEat	SET PF*n*	=
CInsert	Replace	SET PName	?
CLAst	RESet	SET Point	
CLocate	RESTore	SET PREfix	
CMS	RGTLEFT	SET RANge	
CMSG	RIght	SET RECFm	
COMMAND	SAVE	SET REMOte	
COpy	SCHANGE	SET RESERved	
COUnt	SET ALT	SET SCALe	
CP	SET APL	SET SCOPE	
CReplace	SET ARBchar	SET SCReen	
CURsor	SET AUtosave	SET SELect	
DELete	SET BFSLine	SET SERial	
Down	SET BRKkey	SET SHADow	
DUPlicat	SET CASE	SET SIDcode	
EMSG	SET CMDline	SET SPAN	
EXTract	SET COLOR	SET SPILL	
FILE	SET COLPtr	SET STAY	
Find	SET CTLchar	SET STReam	
FINDUp	SET CURLine	SET SYNonym	
FOrward	SET DISPlay	SET TABLine	
GET	SET ENTer	SET TABS	
Help	SET ESCape	SET TERMinal	
Input	SET ETARBCH	SET TEXT	
Join	SET ETMODE	SET TOFEOF	
LEft	SET FILler	SET TRANSLat	
LOAD	SET FMode	SET TRunc	
Locate	SET FName	SET VARblank	
LOWercas	SET FType	SET Verify	
LPrefix	SET FULLread	SET WRap	
MACRO	SET HEX	SET Zone	
MODify	SET IMage	SET =	
MOve	SET IMPcmscp	SHift	
MSG	SET LASTLorc	SI[1]	
Next	SET LINENd	SOS	
NFind	SET LRecl	SPlit	
NFINDUp	SET MACRO	SPLTJOIN	
PARSE	SET MASK	STAck	

Subcommands that Support SET ETMODE

The following subcommands return the ETMODE setting.

```
EXTRACT
MODIFY
QUERY
STATUS
```

Subcommand	Results
EXTRACT /ETMODE/	returns ON or OFF as specified in the SET ETMODE subcommand.

1. SI should not be used in a byte file system (BFS) path name when ETMODE is on.

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

MODIFY ETMODE	displays SET ETMODE ON or SET ETMODE OFF on the command line.
QUERY ETMODE	displays ON or OFF as defined in the SET ETMODE subcommand.
STATUS	returns the setting of ETMODE and the settings of the other SET subcommand options.

Subcommands that Support SET ETARBCH

The following subcommands recognize the ETARBCH setting.

```
EXTRACT
MODIFY
PRESERVE/RESTORE
QUERY
```

Subcommand	Results

EXTRACT /ETARBCH/	returns ON or OFF and the extended arbitrary character specified in the SET ETARBCH subcommand.
	`ETARBCH.0   number of variables returned`
	`       .1   ON│OFF`
	`       .2   extended arbitrary character`
	`             enclosed by a shift-out and`
	`             a shift-in character`

MODIFY ETARBCH	returns SET ETARBCH ON|OFF character on the command line.
PRESERVE	saves the ETARBCH setting until a following RESTORE subcommand is issued.
RESTORE	restores the ETARBCH setting to the value it had when the PRESERVE subcommand was issued.
QUERY ETARBCH	displays ON or OFF and the extended arbitrary character defined in the SET ETARBCH subcommand.

XEDIT Subcommands with SET ETMODE ON

The following subcommands are described in this section.

```
CAppend          Join             SET SIDcode
CDelete          LEft             SET SPILL
CFirst           Locate           SET SYNonym
Change           LOWercas         SET TRunc
CInsert          NFind, NFINDUp   SET Verify
CLAst            PUT, PUTD        SET Zone
CLocate          Replace          SHift
CReplace         SET BFSLine      SPlit
CURsor           SET LRecl        SPLTJOIN
Find, FINDUp     SET NAMetype     SUPerset
GET              SET PName        UPPercas
Input            SET SERial
```

CAPPEND

When you append a DBCS string to another DBCS string, the shift-in and shift-out characters between the two strings are dropped. For example, if you issue

```
CAPPEND  ␎ＢＢ␏
```

against the current line:

```
      |...+....1....+....2....+....3....+....4....
00010 eee␎ＡＡＡＡＡ␏
```

The result is:

```
      <...+....1....|....2....+....3....+....4....
00010 eee␎ＡＡＡＡＡＢＢ␏
```

CDELETE

When you specify a column target with the CDELETE subcommand, that column is checked to see if it is the second byte of a DBCS character. If so, the previous column is considered the target. CDELETE does not delete a shift-out or shift-in character if it is required to maintain the integrity of the DBCS string.

For example, if the column pointer is at column 1 and you issue the subcommand:

```
CDELETE :8
```

against the current line:

```
      |...+....1....+....2....+....3....+....4....
00010 eee␎ＡＢＣＤＥ␏
```

The result is:

```
      |...+....1....+....2....+....3....+....4....
00010 ␎ＢＣＤＥ␏
```

CFIRST

If you issue CFIRST and the left zone is the second byte of a DBCS character, the column pointer is repositioned at the left zone plus one.

CHANGE

When you issue the CHANGE subcommand, DBCS strings can be in the string to be changed and in the new string.

Example:

Subcommand:

C/␎Ａ␏/␎Ｂ␏/	changes	␎Ａ␏	to	␎Ｂ␏
		␎ＡＣＣ␏		␎ＢＣＣ␏
		␎ＣＣＡ␏		␎ＣＣＢ␏
		␎ＣＡＣ␏		␎ＣＢＣ␏
C/␎Ａ␏e/␎Ｂ␏E/	changes	␎Ａ␏eeee	to	␎Ｂ␏Eeee
		␎ＣＡ␏ee		␎ＣＢ␏Ee
	does not change	␎ＣＡＣ␏ee	to	␎ＣＢＣ␏Ee

You can specify an extended arbitrary character (with SET ETARBCH) in a DBCS string in the same fashion as the arbitrary character (SET ARBCHAR) is used in

EBCDIC strings to specify any characters may appear in the matching string in the file. Following are examples of using CHANGE with arbitrary and extended arbitrary characters.

Example:

Subcommand:	**Changes:**	**To:**
`C/e$e/␎Ａ￥Ｃ␏/`	`e␎Ｂ␏e`	`␎ＡＢＣ␏`
`C/e␎￥␏e/␎Ａ␏$␎Ｃ␏/`	`eEe`	`␎Ａ␏E␎Ｃ␏`
`C/e$e/␎Ａ␏$␎Ｃ␏/`	`eE␎Ｂ␏Ee`	`␎Ａ␏E␎Ｂ␏E␎Ｃ␏`
`C/e␎￥␏e/␎Ａ￥Ｃ␏/`	`e␎Ｂ␏E␎Ｂ␏e`	`␎ＡＢ␏E␎ＢＣ␏`
`C/␎Ａ￥Ｃ␏/e$e/`	`␎Ａ␏E␎Ｃ␏`	`eEe`
`C/␎Ａ␏$␎Ｃ␏/e␎￥␏e/`	`␎ＡＢＣ␏`	`e␎Ｂ␏e`
`C/␎Ａ␏$␎Ｃ␏/e$e/`	`␎ＡＢ␏E␎ＢＣ␏`	`e␎Ｂ␏E␎Ｂ␏e`
`C/␎Ａ￥Ｃ␏/e␎￥␏e/`	`␎Ａ␏E␎Ｂ␏E␎Ｃ␏`	`eE␎Ｂ␏Ee`

When a double-byte character is at or beyond the zone boundary, it is not changed.

Example:

Subcommand:

```
                                   ...|<....>....+.         ...|<....>....+.
C/␎Ｂ␏/␎Ｃ␏/     does not change   ␎ＡＢＣＤＥ␏       to   ␎ＡＣＣＤＥ␏

                                   ...|<....>....+.         ...|<....>....+.
C/␎Ｅ␏/␎Ｆ␏/     does not change   ␎ＡＢＣＤＥ␏       to   ␎ＡＢＣＤＦ␏

                                   ...|<....>....+.         ...|<....>....+.
C/␎ＣＤ␏/␎ＡＡ␏/    changes         ␎ＡＢＣＤＥ␏       to   ␎ＡＢＡＡＥ␏
```

CINSERT

When a DBCS string is inserted into another DBCS string, the shift-out and shift-in characters are not inserted. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７␏eee
```

Issuing the subcommand:

```
CINSERT ␎ＡＡ␏
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３ＡＡ４５６７␏eee
```

When an EBCDIC string is inserted into a DBCS string, appropriate shift-in and shift-out characters are inserted to separate the original DBCS string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７␏eee
```

Issuing the subcommand:

```
CINSERT eeee
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３␏eeee␎４５６７␏eee
```

When an DBCS string is inserted into an EBCDIC string, the shift-out and shift-in characters are inserted as part of the string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eeeeeeeeeeeeeeeeeeeeeee
```

Issuing the subcommand:

```
CINSERT ␎ＡＡ␏
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eeeeeeeeee␎ＡＡ␏eeeeeeeeeeee
```

CLAST

When you use the CLAST subcommand to move the column pointer to the end of the zone, a check is made to see if the right zone is at the second byte of a DBCS character. If the right zone is the second byte of a DBCS character, the column pointer is adjusted to the first byte of the DBCS character (to zone right minus one).

CLOCATE

If the column target is set on the second byte of a DBCS character, the column pointer is repositioned to the previous column. If the column pointer is at TOL or EOL, no adjustment occurs.

CREPLACE

When you replace part of a DBCS string with EBCDIC characters, shift-out and shift-in characters are generated to maintain proper pairing of the characters. If the second byte of a DBCS character remains after the subcommand executes, it is replaced by a blank. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７␏eee
```

Issuing the subcommand:

```
CREPLACE eee
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３␏eee ␎７␏eee
```

When you replace one DBCS string with another DBCS string the shift-out and shift-in characters are not used as a part of the replacement string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７␏eee
```

Issuing the subcommand:

```
CREPLACE ␎ＡＡ␏
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３ＡＡ６７␏eee
```

However, when a DBCS string replaces part of an EBCDIC string, the shift-out and shift-in characters are included in the replacement string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２␏eeeeeee␎３４␏eee
```

Issuing the subcommand:

```
CREPLACE ␎Ａ␏
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２␏e␎Ａ␏ee␎３４␏eee
```

CURSOR

The CURSOR subcommand adjusts the cursor position so the character boundaries of DBCS characters are respected. For example (the cursor is at column 1):

```
      <...+....1|...+....2....+....3....+....4....
00010 _eee␎１２３４５６７８９␏eee
```

Issuing the subcommand:

```
CURSOR FILE 10 20
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee␎１２３４５６７８９␏eee
```

EXTRACT

A shift-out character, as the self-defining delimiter in the EXTRACT subcommand, results in a return code of 5. EXTRACT.0 is set to 1 and EXTRACT.1 is set to the nonvalid delimiter (the shift-out character).

FIND, FINDUP

The shift-in character is ignored during a search for a DBCS string. Issuing:

```
FIND ␎１２␏
```

locates either:

␎１２␏ **or** ␎１２３␏

GET

If truncation of a DBCS string occurs when inserting lines from another file, a shift-in character is inserted to close the DBCS string.

INPUT

If truncation of a DBCS string occurs when inserting a line, a shift-in character is inserted to close the DBCS string.

JOIN

When two DBCS strings are joined, the shift-in character from the first string and the shift-out character from the second string are dropped. For example (the column pointer is at column 24):

```
      <...+....1....+....2...|+....3....+....4....
00010 eee␎１２３４５６７８９␏eee
00011 ␎ＡＡＡＡＡＡＡ␏
```

Issuing the subcommand:

```
JOIN COLUMN
```

results in:

```
      <...+....1....+....2..|.+....3....+....4....
00010 eee▯１２３４５６７８９ＡＡＡＡＡＡ▯
```

When an EBCDIC string is joined to a DBCS string, a shift-in character is inserted at the end of the DBCS string. For example (the column pointer is at column 11):

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▯１２３４５６７８９▯eee
00011 EEEEEEEEEEEEE
```

Issuing the subcommand:

```
JOIN COLUMN
```

results in:

```
      <...+....1|...+....2....+....3....+....4....
00010 eee▯１２３▯EEEEEEEEEEEEE
```

LEFT

If issuing a LEFT subcommand results in the display of *negative columns*, you should not type data in these columns or you may lose DBCS strings.

LOCATE

The ability to locate a line by specifying a target is one of the editor's most useful functions. A target is the operand of the LOCATE subcommand and many other XEDIT subcommands. The following XEDIT subcommands accept DBCS strings in targets.

Subcommands that Accept DBCS Strings as Targets

ALL	DELete	REPEat
ALter	DUPlicat	SET RANge
CDelete	EXTract	SET SELect
Change	Locate	SHift
CLocate	LOWercas	STAck
COpy	MOve	Type
COUnt	PUT, PUTD	UPPercas.

Special Considerations for Targets

When XEDIT scans a file for a string target, an SO or SI is ignored in the following cases:

- The first character of a string target is an SO.
- The last character of a string target is an SI.
- The SO or SI is specified immediately preceding or immediately following:
  – An EBCDIC or double-byte blank when VARBLANK is ON
  – An arbitrary character when ARBCHAR is ON
  – An extended arbitrary character when ETMODE and ETARBCH are ON

SET CASE

The IGNORE and RESPECT settings of the SET CASE subcommand have no effect when searching for a DBCS string, because there are no lowercase/uppercase pairs for double-byte characters. For example:

```
SET CASE M IGNORE
/e␎ａ␏e/
```

locates:

```
E␎ａ␏E
```

It does not locate:

```
E␎Ａ␏E
```

SET VARBLANK

You can use the SET VARBLANK ON or OFF subcommand to control whether the number of blank characters is significant when searching for a string. For example:

```
SET VARBLANK ON
/e ␎Ａ␏/
```

locates in the text either of the following:

```
e ␎　␏ ␎Ａ␏
e␎　␏ ␎　Ａ␏
```

SET SPAN

DBCS strings at the end of a line and at the beginning of the next line are treated as one string when SET SPAN is ON. The shift-in character at the end of the first line and the shift-out character at the beginning of the second line are ignored. With SET SPAN ON BLANK, one double-byte blank is inserted between the first and the second string.

Example:

Command	Target	Lines in a File	Match / No Match
SET SPAN ON NOBLANK 2	/␎ＡＢ␏/	00010 ␎１Ａ␏ 00011 ␎Ｂ１␏	Match
	/␎Ａ　Ｂ␏/	00010 ␎１Ａ␏ 00011 ␎Ｂ１␏	No Match
SET SPAN ON BLANK 2	/␎ＡＢ␏/	00010 ␎１Ａ␏ 00011 ␎Ｂ１␏	No Match
	/␎Ａ　Ｂ␏/	00010 ␎１Ａ␏ 00011 ␎Ｂ１␏	Match

SET ETARBCH
An arbitrary character (SET ARBCHAR) in an EBCDIC string specifies any characters can appear in the matching string in the file. An extended arbitrary character (SET ETARBCH) in a DBCS string works the same way. For example:

```
SET ETARBCH ON
/␎Ａ￥Ａ␏/
```

locates in the text any of the following:

```
␎○Ａ○Ａ○␏
␎Ａ␏eee␎Ａ␏
␎Ａ␏e␎○␏e␎Ａ␏
```

SET ZONE
When a double-byte character is at or beyond the zone boundary, it is not located. For example:

/␎２␏/	does not locate	`...|<....>....+....` ␎１２３４５６␏
/␎５␏/	does not locate	`...|<....>....+....` ␎１２３４５６␏
/␎２␏/	locates	`..|<+....1>...+....` ␎１２３４５６␏
/␎５␏/	locates	`..|<+....1>...+....` ␎１２３４５６␏

LOWERCAS

When you issue the LOWERCAS subcommand, DBCS strings are excluded from translation. Double-byte characters have no lowercase or uppercase pairs.

NFIND, NFINDUP

The shift-in character is ignored during a search for a DBCS string. Issuing:

```
FIND ␎ＡＢ␏
```

locates either:

```
␎ＡＢ␏   or   ␎ＡＢＣ␏
```

PUT, PUTD

If you use PUT or PUTD to append data to a fixed format file, DBCS strings may be truncated. You receive a message warning you DBCS strings may have been lost due to truncation.

REPLACE

If a DBCS string is truncated when a line is inserted, a shift-in character is inserted to close the DBCS string.

SET LRECL

If you use SET LRECL to shorten the logical record length of a file being edited, you receive a message warning you that DBCS strings may be lost when the file is saved or filed.

SET SERIAL

If you specify a serial string longer than eight characters, XEDIT recognizes only the first eight characters. When ETMODE is ON and the eighth character of the serial string falls within a DBCS string, the editor adjusts the serial string to maintain the integrity of the DBCS string. For example, if you specify the serial string:

```
ab␎１２３␏
```

XEDIT recognizes it as the following:

```
ab␎１２␏
```

SET SIDCODE

If you specify a sidcode string longer than eight characters, XEDIT recognizes only the first eight characters. When ETMODE is ON and the eighth character of the sidcode string falls within a DBCS string, the editor adjusts the sidcode string to maintain the integrity of the DBCS string. For example, if you specify the sidcode string:

```
ab␎１２３␏
```

XEDIT recognizes it as the following:

```
ab␎１２␏
```

SET SPILL

When spilling occurs within a DBCS string, a shift-in character is inserted at the end of the first line and a shift-out character is inserted at the front of the remaining string that is spilled.

SET SYNONYM

When ETMODE is ON, a synonym name may consist of single-byte characters, double-byte characters, or a combination of single-byte and double-byte characters. However, an abbreviation of a synonym name that includes a DBCS string, must use the complete DBCS string contained in the synonym. For example, the following synonym:

```
a␎１２␏b
```

can be abbreviated as:

```
a  or  ab␎１２␏
```

but cannot be abbreviated as:

```
a␎１␏
```

If you specify a synonym name longer than eight characters, XEDIT recognizes only the first eight characters. When ETMODE is ON and the eighth character of the synonym name falls within a DBCS string, the editor adjusts the synonym name to maintain the integrity of the DBCS string. For example, if you specify the synonym:

```
ab␎１２３␏
```

XEDIT recognizes it as the following:

```
ab␎１２␏
```

SET TRUNC

Certain XEDIT subcommands insert characters in a file line shifting existing data to the right. If a DBCS string is shifted beyond the truncation column (defined in SET TRUNC), a shift-in character is generated to maintain the proper pairing of the characters. For example, issuing the subcommands:

```
SET TRUNC 15
CLOCATE :4
```

on the following string:

```
      <..|+....1....>....2....+....3....+....4....
00010 eee␎ＡＢＣＤ␏
```

Then, issuing the subcommand:

```
CINSERT eeeee
```

results in:

```
      <..|+....1....>....2....+....3....+....4....
00010 eeeeeeee␎ＡＢ␏
```

The last DBCS character before the truncation column is replaced by a shift-in character and a blank.

Other XEDIT subcommands shorten the length of the file line and insert blanks at the truncation column. If the truncation column falls within a DBCS string, blanks are inserted after the shift-in that follows the truncation column. The CDELETE, CREPLACE, CHANGE, and SHIFT subcommands treat the TRUNC setting in this manner. For example, issuing the subcommands:

```
ZONE 11 *
SET TRUNC 20
CLOCATE :17
```

so the truncation column falls within a DBCS string:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eeeeeeeeeeeee␎ＡＢＣＤ␏eeeeeeeeeee
```

Then, issuing the subcommand:

```
CDELETE 2
```

results in:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eeeeeeeeeeeee␎ＡＣＤ␏  eeeeeeeeeee
```

The line shifted starting at the shift-in character instead of the truncation column (column 20 in this example).

SET VERIFY

When ETMODE is OFF, you can issue SET VERIFY to display multiple column pairs. Then, if you issue SET ETMODE ON, only the first pair of verify columns continues to be displayed. However, when ETMODE is ON and you issue SET VERIFY with multiple column pairs on a terminal that supports DBCS, you get an error message.

For example, when ETMODE is OFF and you issue the subcommand:

```
V 1 20 40 50
```

columns 1 through 20 and 40 through 50 are displayed.

Then, if you issue:

```
SET ETMODE ON
```

only columns 1 through 20 continue to be displayed.

When ETMODE is ON and the column(s) specified in SET VERIFY falls within a DBCS string, overtyping to change DBCS characters to EBCDIC characters on the first or last column may change the original character that does not appear on the screen. For example:

```
00010 eeeeeeeeeeeee␎ＡＢＣＤ␏eeeeeeeeeee
      ....+....1....+....2....+....3....+....4....
```

issuing the following:

```
SET VERIFY 15 *
```

results in:

```
00010  ␎ＢＣＤ␏eeeeeeeeeee
      +....2....+....3....+....4....
```

If you overtype the line as follows:

```
00010  a ␎ＣＤ␏eeeeeeeeeee
      +....2....+....3....+....4....
```

you have changed the first two double-byte characters:

```
00010 eeeeeeeeeeeee␎␏a ␎ＣＤ␏eeeeeeeeeee
      |...+....1....+....2....+....3....+....4....
```

SET ZONE

Certain XEDIT subcommands ignore data outside the left and right zone (defined in SET ZONE). If the zone column falls within a DBCS string, the subcommand executes in the entire DBCS string up to and including the shift-in or shift-out character. The CDELETE and SHIFT subcommands treat the ZONE settings in this manner. If the left zone falls within a DBCS string, the shift-out character is treated as the left zone. If the right zone falls within a DBCS string, the shift-in character is treated as the right zone. For example, issuing the subcommands:

```
SET ZONE 11 20
CLOCATE :17
```

so a zone column falls within a DBCS string:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eee␎ＡＢＣＤ␏eeeeeeeeeee
```

Then, issuing the subcommand:

```
SHIFT RIGHT 1
```

results in:

```
      ....+....1<...+.|..>....+....3....+....4....
00010 eee ␎ＡＢＣＤ␏eeeeeeeeeee
```

The line shifted starting at the shift-out character instead of the left zone (column 11 in this example).

SHIFT

When shifting data beyond the file boundary (left of zone left or right of the truncation column), shift-out and shift-in characters are generated to maintain proper pairing of the characters. For example (the column pointer is at column 4):

```
SHIFT LEFT 5
```

Then issuing the subcommand:

```
SHIFT LEFT 5
```

results in:

```
      <..|+....1....+....2....+....3....+....4....
00010 ␎１２３４５６７８９␏eee
```

For another example (column pointer is at column 4, truncation column is at column 25):

```
      <..|+....1....+....2....>....3....+....4....
00010 eee␎１２３４５６７８９␏ee
```

Then issuing the subcommand:

```
SHIFT RIGHT 5
```

results in:

```
      <..|+....1....+....2....>....3....+....4....
00010      eee␎１２３４５６７␏
```

In this example the last DBCS character before the truncation column is replaced by a shift-in character and a blank.

SPLIT

When you split a DBCS string, a shift-in character is inserted at the end of the original line and a shift-out character is inserted at the first column of the new line. For example (column pointer is at column 11):

```
      <...+....1|...+....2....>....3....+....4....
00010 eee␎ＡＢＣＤＥＦＧＨＩ␏ee
```

If you issue the subcommand:

```
SPLIT COLUMN
```

the result is:

```
00010 eee␎ＡＢＣ␏
      <...+....1|...+....2....>....3....+....4....
00011 ␎ＤＥＦＧＨＩ␏ee
```

When the split places all the DBCS characters on a new line the shift-out character is deleted from the original line and inserted at the beginning of the new line. For example (column pointer is at column 5):

```
      <...|....1....+....2....>....3....+....4....
00010 eee␎ＡＢＣＤＥＦＧＨＩ␏ee
```

When you issue the subcommand:

```
SPLIT COLUMN
```

the result is:

```
00010 eee
      <...|....1....+....2....>....3....+....4....
00011 ␎ＡＢＣＤＥＦＧＨＩ␏ee
```

SPLTJOIN

When you split a DBCS string, a shift-in character is inserted at the end of the original line and a shift-out character is inserted at the first column of the new line.

When two DBCS strings are joined, the shift-in character from the first string and the shift-out character from the second string are dropped. When an EBCDIC string is joined to a DBCS string, a shift-in character is inserted at the end of the DBCS string. For examples of splitting and joining lines, see the descriptions of SPLIT and JOIN in this section.

UPPERCAS

When you issue the UPPERCAS subcommand, DBCS strings are excluded from translation. Double-byte characters have no lowercase or uppercase pairs.

Error Message 580E

When ETMODE is ON the following error message may be issued from any XEDIT subcommand or macro:

```
580E Invalid string: [Shift-out (SO) is not a valid
     delimiter | Unmatched shift-out (SO) and shift-in
     (SI) | Odd number of characters between SO and
     SI | Invalid doublebyte character(s)].,RC=5
```

Restrictions for DBCS

The following are the DBCS restrictions:

1. On terminals without DBCS character support, such as a 3277, DBCS data is not displayed correctly.
2. The XEDIT sort algorithm does not apply to DBCS codes.
3. A shift-out character (X'0E') cannot be used as a self-defining delimiter. For example, the following is not valid:

```
LOCATE ␎Ａ␏
```

   You must specify a delimiter, such as a slash (/). For example, this is valid:

```
LOCATE /␎Ａ␏/
```

4. You cannot specify a DBCS string as a file name, file type, or a file mode because of restrictions in the CMS file system.
5. XEDIT does not allow DBCS strings to span lines in a file. The DBCS string must be contained in a single line so shift-out and shift-in pairs are maintained on each line in the file. Consider this when editing a PL/I source statement.
6. Data stacked with the STACK, READ, and TRANSFER subcommands may be truncated because of the CMS stack limitation of 255 characters. When you write macros, remember the data being stacked will be truncated if it exceeds 255 characters.
7. The following subcommands are not allowed in extended mode (SET ETMODE ON). You can execute these subcommands with SET ETMODE OFF. However, DBCS strings are treated as EBCDIC data and you may get unpredictable results about DBCS data.

```
COMPress          HEXType           POWerinp
COVerlay          MErge             SORT
EXPand            Overlay
```

8. When SET ETMODE is ON, XEDIT does not scan for control characters (defined in SET CTLCHAR) within a DBCS string.
9. When a DBCS string is specified in a PARSE subcommand, XEDIT stops searching for the self-defining delimiter within a DBCS string.
10. DBCS strings are treated as EBCDIC data (are not recognized) in byte file system (BFS) path names.

DBCS Strings

Before XEDIT displays any data or accepts any operands, it verifies the DBCS string is valid. A valid DBCS string has the following characteristics.

1. All DBCS strings must be preceded by a shift-out character and followed by a shift-in character.
2. The length of the string must be even.
3. The hex code for a DBCS character must be X'0000', X'4040', or X'*aabb*', with *aa* and *bb* both being in the range of X'41' to X'FE'.

If any of the above conditions are not met when using DBCS strings with an XEDIT subcommand, an error message is issued.

When you are displaying data, if any invalid codes are found, the DBCS character is replaced with a DBCS NONDISP character (X'427F'). Nonvalid DBCS strings are displayed as EBCDIC data. The shift-out and shift-in characters are replaced with a double quote (or the character defined in SET NONDISP) and the remainder of the string is displayed as EBCDIC.

Appendix G. XEDIT Virtual Screens and Windows

General Description

On a display terminal, XEDIT always uses a virtual screen and window. The window and virtual screen have the same name. The name depends on whether you specify the WINDOW option on the XEDIT command. If you choose the window option, XEDIT uses the name specified. If you omit the WINDOW option, XEDIT uses *XEDIT* for the virtual screen and window name.

When editing a file, XEDIT checks to see if the specified virtual screen and window exist. If both do not exist, XEDIT tries to make the window and virtual screen the same size whenever possible. However, windows cannot be larger than the physical screen, and the virtual screen XEDIT uses must have at least 5 lines and 20 columns. Table 7 summarizes how XEDIT handles virtual screen and window setup:

Table 7. How XEDIT Defines Virtual Screens and Windows

Virtual Screen	**Window**	**XEDIT Action**
Does not exist	Does not exist	XEDIT creates window and screen with the same dimensions as the physical screen.
Exists	Does not exist	XEDIT makes the window the same size as the virtual screen if the virtual screen is smaller than the physical screen and has at least 5 lines and 20 columns. If the virtual screen is larger than the physical screen, XEDIT changes lines or columns or both to the size of the physical screen. If the virtual screen is smaller than 5 lines or 20 columns, XEDIT issues an error message.
Does not exist	Exists	XEDIT makes the virtual screen the same size as the window if the window has at least 5 lines and 20 columns. If the window is smaller than 5 lines or 20 columns, XEDIT changes lines or columns or both to the size of the physical screen.
Exists	Exists	XEDIT uses the virtual screen and window if the virtual screen has at least 5 lines and 20 columns. If not, XEDIT issues an error message.

The following examples illustrate some of the conditions described in the preceding table:

- A virtual screen has 24 lines and 80 columns. No window exists. Your terminal has a physical screen with 32 lines and 80 columns. XEDIT defines the window with 24 lines and 80 columns.
- A virtual screen has 100 lines and 200 columns. No window exists. Your terminal has a physical screen with 24 lines and 80 columns. XEDIT defines the window with 24 lines and 80 columns.
- A window has 20 lines and 60 columns. No virtual screen exists. Your terminal has a physical screen with 32 lines and 80 columns. XEDIT defines the virtual screen with 20 lines and 60 columns.
- A virtual screen has 24 lines and 200 columns. No window exists. Your terminal has a physical screen with 32 lines and 80 columns. XEDIT defines the window with 24 lines and 80 columns.

Default Options for the XEDIT Window

In the default situation, that is, when XEDIT defines the window for you, XEDIT uses the following window options:

```
FIXED                         NOTOP
NOBOR                         USER
NOPOP
```

See *z/VM: CMS User's Guide* for a description of these options.

Full-Screen CMS and the CMSOUT Window

While you are in the XEDIT environment, the CMSOUT window may be displayed at various times. If full-screen CMS is on before XEDIT writes to the screen, the CMSOUT window may be displayed followed by the XEDIT window or the particular window that has been set up to display the file.

When you return to the XEDIT environment from CMS subset mode, the CMSOUT window is shown, followed by the window set up to display the file.

If you are already in XEDIT and you set full-screen CMS ON, the CMSOUT window is shown if the CMSOUT window has never been shown in this XEDIT session and you issue a CMS command, or if an XEDIT message is issued that cannot fit into the XEDIT message area.

Entering XEDIT from Full-Screen CMS

Special Considerations: If you enter XEDIT when full-screen CMS is ON and you then run an application or an exec that issues a prompt and causes a VM READ status, the CMSOUT window is displayed with a command line. At the command line, type in your response and press ENTER. When the exec terminates, the CMSOUT window is returned to its original state and is displayed without a command line the next time it is displayed on the screen.

If you enter XEDIT when full-screen CMS is ON and you then run an application or an exec that **does not** issue a prompt but **does** cause a VM READ status, the WM window is displayed and the following message is issued:

```
Active window overlaid; enter a windowing command or press a PF key
```

In this situation, exit from the WM window; you are placed in full-screen CMS and the status line prompts you:

```
Enter your response in vscreen CMS
```

Enter your response and press ENTER. You are again returned to the WM window and receive this message:

```
Active window overlaid; enter a windowing command or press a PF key
```

When you exit from the WM window, you are returned to the XEDIT environment.

Invoking Full-Screen CMS from XEDIT

If you set full-screen CMS ON while you are in the XEDIT environment, the CMS window is displayed and you can enter commands in the WM window. In this case, the XEDIT window is not visible. To make the XEDIT window visible again, use the CMS WINDOW POP command, specifying the window name XEDIT is using.

Similarly, if the active window is not large enough for you to enter a command, you can use the PA1 key, which has a default setting of WINDOW POP WM, to display the WM window. You can then enter CMS commands to adjust the window. For example, you can use the CMS WINDOW MAXIMIZE to make the window larger.

Disconnect/Reconnect Considerations

If, while in XEDIT, you disconnect from your terminal and later reconnect to a different type of terminal, the virtual screen and window XEDIT uses may be resized to fit the dimensions of the new terminal.

The virtual screen and window change as follows:

- If you have **not** defined the virtual screen and window before disconnecting (this is the default situation), after reconnecting:
  1. The virtual screen is redefined according to the procedure described earlier in this appendix.
  2. The window is resized to fit the new physical screen.
- If you have defined a virtual screen before disconnecting, no adjustments are made to it after reconnecting.
- If you have defined a window before disconnecting, and you reconnect to a **smaller** screen, XEDIT adjusts the window to fit on the new screen. No adjustment occurs when reconnecting to an equivalent or larger screen.

Note: If you define the XEDIT virtual screen before disconnecting and reconnect onto a smaller screen, the XEDIT window may be adjusted such that the resulting window is smaller than the virtual screen. In such a case, the command line may not be visible in the window. You can use the PA1 key, which has a default setting of WINDOW POP WM, to display the WM window. You can then use the CMS SCROLL command to position the command line in the window.

If you have an exec that disconnects you, the exec will continue to run while the virtual machine is disconnected. If the XEDIT command is issued while the virtual machine is disconnected, the XEDIT session will be in line mode when you reconnect. If full-screen is on, the full-screen XEDIT session will be displayed when you reconnect.

Usage Notes

1. On leaving XEDIT, any virtual screens or windows XEDIT defined, including the default XEDIT virtual screen and window, are deleted. Only user-defined virtual screens and windows remain. XEDIT also clears any virtual screens and hides any windows it used but did not create.

   Note that if full-screen CMS is ON and there are multiple XEDIT sessions, the CMSOUT window is hidden only after leaving the last one.
2. If, during an XEDIT session, you delete the virtual screen XEDIT is using, XEDIT redefines the virtual screen according to the guidelines explained in Table 7 on page 487.
3. When XEDIT is using or has used a virtual screen, the following CMS commands are not valid for that virtual screen:
   ```
   VSCREEN CLEAR                     SET LOGFILE
   VSCREEN CURSOR                    SET VSCREEN
   VSCREEN GET                       VSCREEN WAITREAD
   VSCREEN PUT                       VSCREEN WAITT
   VSCREEN ROUTE                     VSCREEN WRITE
   ```
   Also, when XEDIT uses a virtual screen, it uses the screen area settings (PRotect/NOProtect, High/NOHigh, *color*, *exthi*, PS*s*) defined in XEDIT SET subcommands. These XEDIT-defined settings override any settings that may have been specified on the CMS VSCREEN DEFINE command.

4. If you specify the WINDOW option on the XEDIT command and XEDIT uses an existing virtual screen, the virtual screen settings change as follows:
   a. The virtual screen TYPE/NOTYPE setting is set to TYPE.
   b. If logging has been specified for that virtual screen, it is set to OFF.
   c. If full-screen CMS is ON, any message classes routed to that virtual screen are rerouted to the CMS virtual screen.
5. It is possible in certain situations to position your window so the command line is not visible in the window. To avoid problems, you should establish a method (setting a PF key, for example) to make the command line visible again.
6. If full-screen CMS is ON and abend processing occurs while you are in XEDIT, the CMS window is displayed with the CMSOUT window popped on top of it. Entering a command hides the CMSOUT window if it is of type SYSTEM or deletes it if it is of type USER.

Working in Line-mode

If the XEDIT terminal setting is TYPEWRITER, output is written a line at a time to CMS. CMS displays the output based on the terminal type and the full-screen setting.

Under certain circumstances, when you are using the editor in typewriter mode on a display terminal, the CMSOUT window may appear. For example, suppose you are in typewriter mode and you invoke a separate XEDIT session with the CMS XEDIT command, set full-screen CMS ON during that XEDIT session. Upon exit from full-screen CMS, the CMSOUT window may remain on your screen even though you are now editing in typewriter mode.

Appendix H. A Summary by Function of XEDIT Subcommands and Macros

The following chart lists XEDIT subcommands by the tasks you may want to do.

Start and Stop Editing, Control Screen, Files, Keys		
Control file characteristics	SET BFSLine	Control the translation between records and byte stream for BFS files.
	SET FMode	Change the file mode of the current file.
	SET FName	Change the file name of the current file.
	SET FType	Change the file type of the current file.
	SET LRecl	Define new logical record length.
	SET NAMetype	File IDs are in CMS format (*fn ft fm*) or in BFS format (*pathname*).
	SET PACK	Control whether the editor packs a file when writing it to a disk.
	SET PName	Change the BFS path name of the current file.
	SET RECFm	Define record format: fixed, variable, fixed packed, variable packed.
	SET SIDcode	Insert string into every line of the update file.
	SET TRunc	Define truncation column (last column for data) as the *n*th column or * (logical record length).
Control keys	SET APL	Inform the editor and CMS whether APL characters are to be used.
	SET BRKkey	Control whether CP breaks in when BRKKEY is pressed.
	SET ENTer	Define or remove a meaning for the ENTER key.
	SET PA*n*	Define or remove a meaning for a hardware attention key.
	SET PF*n*	Define or remove a meaning for a hardware program function (PF) key.
	SET TEXT	Inform the editor and CMS whether text characters are to be used.
Control Message Display	CMSG	Display a message on the command line.
	EMSG	Display a message in the message area and possibly sound the alarm.
	MSG	Display a message in the message area.
	SET MSGLine	Define position of message line and the number of lines a message may expand to.
	SET MSGMode	Control whether messages are displayed and whether all messages are full length.
	SET Verify	Control whether changed lines are displayed (in the message area) and optionally control whether data is displayed in hexadecimal or EBCDIC and what columns are displayed.

Control screen layout	SCALE (prefix)	Display scale at line where entered.
	SET COLOR	Specify colors for various screen fields and, optionally, highlighting and other values. Note, this option may be specified as COLOR or COLOUR.
	SET CMDline	Control the position of the command line.
	SET CURLine	Define position of the current line on the screen.
	SET MASK	Control contents of mask (characters that prefill the new line).
	SET MSGLine	Define position of message line and the number of lines a message may expand to.
	SET NUMber	Control whether line numbers are displayed in the prefix area.
	SET PREfix	Control display of the prefix area or define a synonym for the prefix subcommand.
	SET SCALe	Control the position of the scale.
	SET SCReen	Controls how many horizontal or vertical screens are displayed; you can edit multiple files or multiple views of the same file.
	SET TABLine	Control the position of the tabline (showing a *T* at each tab stop).
	SET TERMinal	Control whether the terminal is used in line mode or full-screen mode.
	SET TOFEOF	Control whether notices (Top of File, End of File, Top of Range, End of Range) are displayed.
	TABL (prefix)	Display the tabline (showing a *T* at each tab stop) at the line where entered.
	SET Verify	Control whether changed lines are displayed (in the message area) and optionally control whether data is displayed in hexadecimal or EBCDIC and what columns are displayed.
Edit new or existing file	Xedit (CMS command)	Edit new or existing file.
	Xedit (subcommand)	Edit another file.
End editing session	CANCEL	Stop editing all files. If the QUIT subcommand is defined to perform a protected QUIT, for changed file(s) you receive warning message(s) and must enter QQuit to quit without saving changes.
	FILE	Stop editing, saving all changes.
	PF3	Initial setting: End editing session. (Same as QUIT)
	QUIT	End editing session of the current file. If the QUIT subcommand is defined to perform a protected QUIT, for changed file(s) you receive warning message(s) and must enter QQuit to quit without saving changes.
Get Help	Help	Display online help menu.
	PF1	Initial setting: Display online help menu.
Save file on disk; stay in edit	SAVE	Write the file on disk (saving the file with all changes) without ending the editing session.
	SET AUtosave	Controls the number of alterations required before the editor automatically saves the file.

Transmission of data	PA2	Initially assigned as the NULL key.
	SET FULLread	Control whether XEDIT and CMS recognize 3270 null characters in the middle of screen lines.
	SET NONDisp	Define a character to XEDIT and CMS that is used in place of nondisplayable characters.
	SET NULls	Control whether trailing blanks are written to the screen as blanks or as nulls.
	SET REMote	Control whether XEDIT removes nulls and combines some data to compress it.
	with insert key	You can use any key defined as NULL key instead of entering SET NULLS ON.

Editing Commands		
Add text into lines	CAppend	Add text at the end of the current line.
	CInsert	Insert text in the current line before the column pointer. (Position the column pointer with CLocate before inserting text.)
	PA2	Allow use of the insert key on an IBM 3270 keyboard to insert characters in a line.
	SET NULls ON	Allow use of the insert key on an IBM 3270 keyboard to insert characters in a line.
Add lines	A (prefix)	Add one or more lines after the line where entered.
	Add	Add one or more lines after the current line.
	I (prefix)	Add one or more lines after the line where entered.
	Input	Insert one specified line after the current line or enter input mode.
	PF2	Initial setting: Add a line after the line pointed to by the cursor.
	POWerinp	After the current line, enter power input mode (continuous typing – enter text without regard for line breaks).
	SI	(Structured Input) After the current line, continuously add lines, automatically indenting to align.
	SI (prefix)	(Structured Input) Continuously add lines, automatically indenting to align.
Change case of letters	LOWercas	Change uppercase to lowercase, from current line to optional target.
	SET CASE	Control whether characters you enter are translated to uppercase or left as mixed case and whether capitalization affects searches (Respect) or not (Ignore).
	SET TRANSLat	Control uppercase translation of specified characters (when keyboards support some non-English characters).
	UPPercas	Change lowercase to uppercase from the current line to an optional target.
Change data	ALter	Change a single character in one or more lines to another (character or hex).
	Change	Change a string in one or more lines. Strings need *not* be equal length. If new string is omitted, a null string is assumed.
	COVerlay	Replace character(s) one for one in the current line, starting at the column pointer. Blanks in the text you enter do *not* overlay data.
	CReplace	Replace character(s) one for one in the current line, starting at the column pointer. Blanks in the text you enter *do* overlay data.
	Overlay	Replace character(s), starting at column one or at the first tab column (if SET IMAGE ON is in effect). Blanks in the text you enter do *not* overlay data.
	PF6	During SChange (Selective Change), press PF5 to locate the string and then PF6 to change the string.
	Replace	Replace the current line with specified text, or delete the current line and enter input mode.
	SCHANGE	(Selective Change) After a Change or a CLocate subcommand, find (with PF5) occurrences of a string and selectively change (with PF6) them throughout a file.

Column pointer commands	CAppend	Add text at the end of the current line.
	CDelete	Delete character(s) from the column pointer to an optional column-target.
	CFirst	Move the column pointer to the beginning of the line (or zone).
	CInsert	Insert text in the current line before the column pointer.
	CLAst	Move the column pointer to the end of the line (or zone).
	CLocate	Move the column pointer to the column-target.
	COVerlay	Replace character(s) one for one in the current line, starting at the column pointer. Blanks you enter *do not* overlay data.
	CReplace	Replace character(s) one for one in the current line, starting at the column pointer. Blanks you enter *do* overlay data.
Copy lines to another place in same file	C, CC (prefix)	Copy line(s) to another place in the same file. Position copied lines with the P or F prefix subcommand.
	COpy	Copy line(s), starting with the current line, to a specified location in the same file.
	F (prefix)	With C (prefix), position copied line(s) *following* the line where entered.
	P (prefix)	With C (prefix), position copied line(s) *preceding* the line where entered.
Copy lines under original lines	DUPlicat	Duplicate line(s) one or more times under original line(s).
	”, ” ” (prefix)	Duplicate line(s) one or more times under original line(s).
Delete or recover deleted data	CDelete	Delete character(s) from the column pointer to an optional column target.
	D, DD (prefix)	Delete one or more lines where entered.
	DELete	Delete line(s) from the current line to an optional target.
	Replace	Delete the current line and enter input mode.
	RECover	Recover deleted line(s), inserting them before the current line.
Double-Byte Character Set (DBCS) data	SET ETARBCH	Define an extended arbitrary character for use in Double-Byte Character strings.
	SET ETMODE	Control whether XEDIT recognizes DBCS strings.
Exclude Lines or Display Subset of Lines	ALL	Display a collection of lines containing a specified target for editing; excludes other lines.
	S (prefix)	Show line(s) you have excluded with the X prefix macro.
	SET DISPlay	Display set(s) of lines having the same selection level (set with SET SELect).
	SET RANge	Define a new top and bottom for file. Only lines within the range are displayed and subcommands (except FILE and SAVE) affect only these.
	SET SELect	Assign a selection level to line(s).
	SET SHADow	Control whether *shadow lines* (reporting number of lines not shown) are displayed.
	Type	Display line(s), from the current line to the target.
	X, XX (prefix)	Exclude one or more lines from display.

Hexadecimal related	ALter	Change a single character in one or more lines to another (character or hex) that may not be available on your keyboard.
	HEXType	Display line(s) from the current line to an optional target in hex and EBCDIC.
	SET HEX	Control whether you can specify targets and string operands in hexadecimal.
	SET Verify	Control whether changed lines are displayed (in the message area) and optionally control whether data is displayed in hexadecimal or EBCDIC and what columns are displayed.
Invoke input mode	Input	Insert a specified line after the current line or enter input mode.
	POWerinp	Start continuous typing input mode after the current line — you can enter text without regard for line breaks.
	Replace	Replace the current line with a specified line, or delete the current line and enter input mode.
	SI	(Structured Input) After the current line, continuously add lines, automatically indenting to align.
	SI (prefix)	(Structured Input) Continuously add lines, automatically indenting to align.
Move column pointer	CFirst	Move the column pointer to the beginning of the line (or zone).
	CLAst	Move the column pointer to the last position of the line (or zone).
	CLocate	Search for the column-target, starting at the column pointer, and move the column pointer (and line pointer, if SET STREAM ON) to the target.
Move cursor	CURsor	Move the cursor to a specified position on the screen and optionally assign a priority to this position.
	PF12	Initial setting: Move cursor from the file line to the command line or from the command line to the last position it was in the file.
Move data horizontally on the screen	COMPress	Replace blank (or filler) characters with tabs from the current line to an optional target; prepares line(s) to move horizontally with EXPAND.
	EXPand	Position data in new tab columns without retyping.
	SHift	Move data left or right. Data loss is possible.
	<, << (prefix)	Shift one line or a block of lines one or more columns to the left.
	>, >> (prefix)	Shift one line or a block of lines one or more columns to the right.
Move lines to another place in file	F (prefix)	With M (prefix), position the moved line(s) *following* the line where entered.
	M, MM (prefix)	Move one line or a block of lines to another position in the file. Position the moved line(s) with the P or F prefix subcommand.
	MOve	Move line(s), starting at the current line, to a new position in the file.
	P (prefix)	With M (prefix), position moved line(s) *preceding* the line where entered.

Move screen horizontally	LEft	Display data a specified number of columns to the left of column one.
	PF10	Initial setting: Display data in columns further to the right or left than currently shown on the screen. Press the key again to return to the original view.
	RGTLEFT	Display data in columns further to the right or left than currently shown on the screen. Reenter the command to return to the original view.
	RIght	Display data a specified number of columns to the right of the last column shown on the screen.
	SET Verify	Control whether changed lines are displayed (in the message area) and optionally control whether data is displayed in hexadecimal or EBCDIC and what columns are displayed.
Move screen vertically or move line pointer	BAckward	Scroll one, an optional number, or * screens toward Top of file.
	Bottom	Scroll to bottom of file (or range).
	Down	Move line pointer a specified number of lines toward End of File.
	FOrward	Scroll one, a specified number, or * screens toward End of file.
	Next	Move line pointer *n* lines toward End of File.
	PF7	Initial setting: Scroll one screen toward Top of file.
	PF8	Initial setting: Scroll one screen toward End of file.
	TOP	Scroll to Top of File line (or top of range).
	Up	Move line pointer a specified number of lines toward Top of File.
	/ (slash) (prefix)	Make line where entered the current line (and optionally move the column pointer under the column number specified).
	all targets	Targets are included in various subcommands, such as Locate.
Name a line	SET Point	Define or remove symbolic name for current line.
	.xxxx (prefix)	Assign *.xxxx* as a symbolic name at the line where entered (and delete *.xxxx* as a symbolic name for another line if already defined).
Number file lines	RENum	Renumber line numbers of VSBASIC or FREEFORT files.
	SET SERial	Control whether serial identification is added to the file and the format of serial identification.

Prefix subcommands, macros and canceling them	A	Add one or more lines after the line where entered.
	C	Copy line(s) to another place in the same file.
	D	Delete one or more lines where entered.
	E	Extend a logical line by one or more virtual lines where entered.
	F	Position moved or copied line(s) *following* the line where entered.
	I	Add one or more lines after the line where entered.
	M	Move one line or a block of lines to another position in the file.
	P	Position moved or copied line(s) *preceding* the lines where entered.
	RESet (entered at command line)	Cancel pending prefix subcommand or macro.
	S	Show line(s) you have excluded (for example, with the X prefix macro).
	SCALE	Display the scale at the line where entered.
	SI	(Structured Input) Continuously add lines, automatically indenting to align.
	TABL	Display the tabline (showing a *T* at each tab stop) at the line where entered.
	X	Exclude one or more lines from display.
	.xxxx	Assign *.xxxx* as a symbolic name at the line where entered (and delete *.xxxx* as a symbolic name for another line, if already defined).
	<	Shift one line or a block of lines one or more columns to the left.
	/ (slash)	Make the line where entered the current line (and optionally move the column pointer under the column number specified).
	>	Shift one line or a block of lines one or more columns right.
	″ (double quote)	Duplicate line(s) one or more times under original line(s).
Repeat or retrieve command	PF6	Initial setting: Display previously entered commands at the command line. Press ENTER to reexecute or type over to change and then press ENTER.
	PF9	Initial setting: Reexecute the last entered subcommand, macro, or command.
	REPEat	Advance line pointer and reexecute the last subcommand.
	&(Ampersand)	Enter before the subcommand to keep the subcommand displayed at the command line. Press ENTER to reexecute or type over to change and then press ENTER.
	= (Equal sign)	Reexecute the last subcommand entered, or, with an optional subcommand, execute a specified subcommand and then reexecute the last one.
	? (Question Mark)	Redisplay the last entered command at the command line. Press ENTER to reexecute or type over to change and then ENTER.

Search for data	ALL	Display all lines containing a specified target and only those lines.
	CLocate	Search for column-target, starting at the column pointer.
	Find	Search for the line that starts with a specified text. Only nonblank characters specified are checked against the file.
	FINDUp	Search backward for the line that starts with a specified text. Only nonblanks in a specified text are checked against the file.
	Locate	Search for a target, starting with the line after the current line. With an optional subcommand operand, it executes the subcommand after finding the target.
	NFind	Search for the line that does not start with a specified text. Only nonblanks in specified text are checked against the file.
	NFINDUp	Search backward for the line that does not start with a specified text. Only nonblanks in a specified text are checked against the file.
	PF5	Initial setting: Locate a string for which you have typed a CLocate or Change command (part of Selective Change).
Search related	COUnt	Count the number of times a string occurs in a file, starting at the current line.
	SET ARBchar	Turn on and optionally define or turn off an arbitrary character so you can specify it in a target definition for a string.
	SET CASE	Control whether characters you enter are translated to uppercase or left mixed case and whether capitalization affects searches (Respect) or not (Ignore).
	SET HEX	Control whether you can specify targets and string operands in hexadecimal.
	SET IMage	Control how the editor treats tabs and backspaces.
	SET RANge	Define a new top and bottom for file, displaying only lines within this range; commands operate only on these lines.
	SET SCOPE	Control whether the editor operates on all lines or on only those displayed (when displaying a subset of lines, as with ALL).
	SET SPAN	Control whether a string target must be in one line to be found or can span lines and optionally whether blank(s) separate consecutive lines temporarily concatenated during a search.
	SET STAY	Control whether the line pointer moves when the search is unsuccessful. (With SET WRAP ON, the line pointer does not move after an unsuccessful search.)
	SET STReam	Control whether the editor searches the entire file or only the current line for column target.
	SET VARblank	Control whether the number of blanks between words is significant during a search.
	SET WRap	Control whether the editor *wraps around* the file if the end of file or range is reached during search.
	SET Zone	Define starting and ending columns scanned during a search.
SET operands	MODify	Display SET subcommand and current operand values at the command line. To change: type over values and press ENTER.
	Query	Display current values of editing options in the message area.
	SET	Change settings of various editing options.
	STATus	Display SET subcommand options and current settings or create a macro containing these settings.

Sort	SORT	Sort from current line to a target, in ascending or descending order, according to specified sort columns.
Split or join lines	Join	Combine lines at the column pointer, cursor, or specified position, or insert a string before joining lines. Optionally remove leading blanks from the line being joined.
	MErge	Combine two sets of lines (side by side).
	PF11	Initial setting: Split a line at the cursor or join two lines if the cursor is at the end of the text.
	SPlit	Split line: optionally at column pointer, at cursor, or at specified position(s). Optionally, add leading blanks to a new line matching those in the original line.
	SPLTJOIN	Split a line at the cursor or join two lines if the cursor is at the end of the text.
Tab related	COMPress	Replace blank (or filler) characters with tabs from the current line to an optional target; prepares line(s) to move horizontally with EXPAND.
	EXPand	Position data in new tab columns without retyping.
	MODify TABS	Display at the command line the SET TABS subcommand and columns of tab stops. To change: type over settings and press ENTER.
	PF4	Initial setting: Tab key.
	SET ENTer TABKEY	You can define the ENTER key as the TAB key or remove a previously defined meaning for the ENTER key.
	SET FILler	Define the filler character the editor uses when you issue EXPAND.
	SET IMage	Control tabs and backspaces.
	SET PA*n* TABKEY	You can define a PA key as the TAB key or remove a previously defined meaning for a PA key.
	SET PF*n* TABKEY	You can define a PF key as the TAB key or remove a previously defined meaning for a PA key.
	SET TABLine	Control the display of the tabline. (The tabline depicts a *T* at each tab stop.)
	SET TABS	Define logical tab stops.
	TABL (prefix)	Display tabline (depicting a *T* at each tab stop) at the line where entered.
Transfer data between files	GET	Copy all or part of another file after the current line.
	PUT	Store line(s) in a temporary file (until inserted to another file with GET) or into a new or existing file. Lines are put at the *end* of the existing file.
	PUTD	Copy line(s) to another file, *deleting them* from the original file. Lines are copied into a temporary file (until added to a file with GET) or into a new or existing file.
Truncation related	SET SPILL	Control whether data is truncated or spilled onto new lines after certain subcommands and whether the editor splits a word between lines if spilled.
	SET TRUNC	Define truncation column (last column for data) as the *n*th column or * (logical record length).

Identify or Issue Commands		
Identify subcommand, macro, synonym	COMMAND	Execute a subcommand without checking for a macro or synonym.
	MACRO	Execute as a macro without checking for a subcommand or synonym.
	SET LINENd	Control whether the editor recognizes a character as line-end and which character is recognized.
	SET MACRO	Control the order in which the editor searches for subcommands and macros.
	SET PREFIX Synonym	Assign a synonym for a prefix subcommand.
	SET SYNonym	Assign a synonym or control whether the editor looks for synonyms.
Issue command to CMS or CP	CMS	Issue a command to CMS for execution or have the editor enter CMS subset mode.
	CP	Issue a command to CP for execution.
	SET IMPcmscp	Control whether XEDIT passes unrecognized subcommands to CMS and CP for execution.

Macro Commands		
Used in macro writing	CMSG	Display a message on the command line.
	EMSG	Display a message in the message area and possibly sound the alarm.
	EXTract	Get information about internal XEDIT variables or file data.
	LOAD	Read a file into storage from the profile.
	MSG	Display a message in the message area.
	PARSE	Scan a stacked line of macro to check the format of its operands.
	PREServe	Save various XEDIT settings. You can then change settings and later return them to what they were before with RESTore.
	PURge	Remove a macro from virtual storage.
	READ	Put data in a console stack (LIFO) from the terminal.
	REFRESH	Display the screen as of the current moment in processing.
	RESTore	Change settings back to what they were when you entered PREServe.
	SET ALT	Change alteration count (in the top line of file information).
	SET CTLchar	Define a control character that specifies color, highlighting, and other features for parts of a reserved line.
	SET LASTLorc	Define contents of the LASTLORC (last locate or change) buffer.
	SET PENDing	Control execution of a prefix macro and status of the screen during execution.
	SET RESERved	Reserve a specified line the editor cannot use or release a reserved line.
	SET =	Put a string (subcommand or macro) into equal (=) buffer. (You can then execute this subcommand with the = subcommand.)
	SOS	Specify functions for screen operation simulation.
	STAck	Put all or part of line(s) from the current line to optional target in console stack.
	STATus	Display SET subcommand options and settings or create a macro containing these settings.
	SUPerset	Improves performance by allowing multiple SET options to be specified in the same subcommand.
	TRAnsfer	Put specified editing variables in a console stack for use by a macro.

Typewriter Terminal Commands		
Used on typewriter terminals	LPrefix	Simulate writing in the prefix area of the current line.
	SET COLPtr	Control whether the column pointer is displayed.
	SET ESCape	Define a character that lets you enter subcommands while in input mode.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, New York 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

Trademarks

Glossary

For a list of z/VM terms and their definitions, see the *z/VM: Glossary* book.

The glossary is also available through the online HELP Facility. For example, to display the definition of “cms”, enter:

```
help glossary cms
```

You will enter the glossary HELP file and the definition of “cms” will be displayed as the current line. While you are in the glossary HELP file, you can also search for other terms.

If you are unfamiliar with the HELP Facility, you can enter:

```
help
```

to display the main HELP menu, or enter:

```
help cms help
```

for information about the HELP command.

For more information about the HELP Facility, see the *z/VM: CMS User’s Guide*.

Bibliography

This bibliography lists the books in the z/VM product library. For abstracts of these books and information about current editions and available media, see *z/VM: General Information*.

Where to Get z/VM Books

z/VM books are available from the following sources:

- IBM Publications Center at www.ibm.com/shop/publications/order/
- z/VM Internet Library at www.ibm.com/eserver/zseries/zvm/library/
- *IBM eServer zSeries Online Library: z/VM Collection* CD-ROM, SK2T-2067

z/VM Base Library

The following books describe the facilities included in the z/VM base product.

System Overview

z/VM: General Information, GC24-6095

z/VM: Glossary, GC24-6097

z/VM: License Information, GC24-6102

z/VM: Migration Guide, GC24-6103

Installation and Service

z/VM: Guide for Automated Installation and Service, GC24-6099

z/VM: Service Guide, GC24-6117

z/VM: VMSES/E Introduction and Reference, GC24-6130

Planning and Administration

z/VM: CMS File Pool Planning, Administration, and Operation, SC24-6074

z/VM: CMS Planning and Administration, SC24-6078

z/VM: Connectivity, SC24-6080

z/VM: CP Planning and Administration, SC24-6083

z/VM: Getting Started with Linux on zSeries, SC24-6096

z/VM: Group Control System, SC24-6098

z/VM: I/O Configuration, SC24-6100

z/VM: Performance, SC24-6109

z/VM: Running Guest Operating Systems, SC24-6115

z/VM: Saved Segments Planning and Administration, SC24-6116

z/VM: Secure Configuration Guide, SC24-6138

z/VM: TCP/IP Planning and Customization, SC24-6125

eServer zSeries 900: Planning for the Open Systems Adapter-2 Feature, GA22-7477

eServer zSeries: Open Systems Adapter-Express Customer's Guide and Reference, SA22-7935

eServer zSeries: Open Systems Adapter-Express Integrated Console Controller User's Guide, SA22-7990

z/OS and z/VM: Hardware Configuration Manager User's Guide, SC33-7989

Customization

z/VM: CP Exit Customization, SC24-6082

Operation

z/VM: System Operation, SC24-6121

z/VM: Virtual Machine Operation, SC24-6128

Application Programming

z/VM: CMS Application Development Guide, SC24-6069

z/VM: CMS Application Development Guide for Assembler, SC24-6070

z/VM: CMS Application Multitasking, SC24-6071

z/VM: CMS Callable Services Reference, SC24-6072

z/VM: CMS Macros and Functions Reference, SC24-6075

z/VM: CP Programming Services, SC24-6084

z/VM: CPI Communications User's Guide, SC24-6085

z/VM: Enterprise Systems Architecture/Extended Configuration Principles of Operation, SC24-6094

z/VM: Language Environment User's Guide, SC24-6101

z/VM: OpenExtensions Advanced Application Programming Tools, SC24-6104

z/VM: OpenExtensions Callable Services Reference, SC24-6105

z/VM: OpenExtensions Commands Reference, SC24-6106

z/VM: OpenExtensions POSIX Conformance Document, GC24-6107

z/VM: OpenExtensions User's Guide, SC24-6108

z/VM: Program Management Binder for CMS, SC24-6110

z/VM: Reusable Server Kernel Programmer's Guide and Reference, SC24-6112

z/VM: REXX/VM Reference, SC24-6113

z/VM: REXX/VM User's Guide, SC24-6114

z/VM: Systems Management Application Programming, SC24-6122

z/VM: TCP/IP Programmer's Reference, SC24-6126

Common Programming Interface Communications Reference, SC26-4399

Common Programming Interface Resource Recovery Reference, SC31-6821

OS/390: DFSMS Program Management, SC27-0806

z/OS: Language Environment Concepts Guide, SA22-7567

z/OS: Language Environment Debugging Guide, GA22-7560

z/OS: Language Environment Programming Guide, SA22-7561

z/OS: Language Environment Programming Reference, SA22-7562

z/OS: Language Environment Run-Time Messages, SA22-7566

z/OS: Language Environment Writing ILC Applications, SA22-7563

End Use

z/VM: CMS Commands and Utilities Reference, SC24-6073

z/VM: CMS Pipelines Reference, SC24-6076

z/VM: CMS Pipelines User's Guide, SC24-6077

z/VM: CMS Primer, SC24-6137

z/VM: CMS User's Guide, SC24-6079

z/VM: CP Commands and Utilities Reference, SC24-6081

z/VM: Quick Reference, SC24-6111

z/VM: TCP/IP User's Guide, SC24-6127

z/VM: XEDIT Commands and Macros Reference, SC24-6131

z/VM: XEDIT User's Guide, SC24-6132

CMS/TSO Pipelines Author's Edition, SL26-0018

Diagnosis

z/VM: Diagnosis Guide, GC24-6092

z/VM: Dump Viewing Facility, GC24-6093

z/VM: System Messages and Codes - AVS, Dump Viewing Facility, GCS, TSAF, and VMSES/E, GC24-6120

z/VM: System Messages and Codes - CMS and REXX/VM, GC24-6118

z/VM: System Messages and Codes - CP, GC24-6119

z/VM: TCP/IP Diagnosis Guide, GC24-6123

z/VM: TCP/IP Messages and Codes, GC24-6124

z/VM: VM Dump Tool, GC24-6129

z/OS and z/VM: Hardware Configuration Definition Messages, SC33-7986

Books for z/VM Optional Features

The following books describe the optional features of z/VM.

Data Facility Storage Management Subsystem for VM

z/VM: DFSMS/VM Customization, SC24-6086

z/VM: DFSMS/VM Diagnosis Guide, GC24-6087

z/VM: DFSMS/VM Messages and Codes, GC24-6088

z/VM: DFSMS/VM Planning Guide, SC24-6089

z/VM: DFSMS/VM Removable Media Services, SC24-6090

z/VM: DFSMS/VM Storage Administration, SC24-6091

Directory Maintenance Facility

z/VM: Directory Maintenance Facility Commands Reference, SC24-6133

z/VM: Directory Maintenance Facility Messages, GC24-6134

z/VM: Directory Maintenance Facility Tailoring and Administration Guide, SC24-6135

Performance Toolkit for VM™

z/VM: Performance Toolkit, SC24-6136

Resource Access Control Facility

External Security Interface (RACROUTE) Macro Reference for MVS and VM, GC28-1366

Resource Access Control Facility: Auditor's Guide, SC28-1342

Resource Access Control Facility: Command Language Reference, SC28-0733

Resource Access Control Facility: Diagnosis Guide, GY28-1016

Resource Access Control Facility: General Information, GC28-0722

Resource Access Control Facility: General User's Guide, SC28-1341

Resource Access Control Facility: Macros and Interfaces, SC28-1345

Resource Access Control Facility: Messages and Codes, SC38-1014

Resource Access Control Facility: Migration and Planning, GC23-3054

Resource Access Control Facility: Security Administrator's Guide, SC28-1340

Resource Access Control Facility: System Programmer's Guide, SC28-1343

Index

Readers' Comments — We'd Like to Hear from You

z/VM
XEDIT Commands and Macros
Reference
Version 5 Release 1.0

Publication No. SC24-6131-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC24-6131-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, New York 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

SC24-6131-00

Cut or Fold
Along Line

Program Number: 5741-A05

Printed in USA

SC24-6131-00

Spine information:

IBM z/VM XEDIT Commands and Macros Reference Version 5 Release 1.0