60455730

CONTROL DATA CORPORATION

XEDIT VERSION 3 REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 1

XEDIT COMMAND SUMMARY

Command	Page	Command	Page
ADD (A)	5-1	MODIFY (M)	5-6
ADDLN (ALN)	7-1	NEXT (N)	4-3
ADDLNS (ALNS)	7-1	NOBELLS (NB)	3-3
BOTTOM (B)	4-3	OCTCHANGE (OC)	8-2
BRIEF (BR)	3-2	PRINT (P)	4-4
CHANGE (C)	5-1	QMOD (QM)	5-6
CHANGES (CS)	5-1	QUIT (Q)	12-1
COPY	9-1	READ	9-3
COPYD	9-2	READP	9-2
(CR)	6-4	REPLACE (R)	6-3
DBADL (DBL)	7-4	REPLACELN (RLN)	7-3
DEFTAB (DT)	11-1	RESTORE (REST)	8-2
DELETE (D)	6-1	RMARGIN (RM)	11-2
DELETELN (DLN)	7-1	STOP	12-1
DELIMIT (DEL)	10-1	TABS (TAB)	11-1
DEOF (DF)	8-3	TEOF	3-3
DEOR (DR)	8-3	TEOR	3-3
DLBLANKS (DLB)	8-1	TOP (T)	4-3
eEDIT	6-5	TOPNULL (TN)	8-1
END (E)	12-1	TRIM	5-4
EXPLAIN	3-4	TRUNCATE (TRUNC)	11-3
FBADL (FBL)	7-3	VERIFY (V)	3-2
FILE (F)	9-3	WEOF (WF)	8-4
FINDLL (FLL)	11-2	WEOR (WR)	8-4
HELP (H)	3-4	WHERE	4-4
INPUTe	6-5	WMARGIN (WM)	5-4
INSERT (I)	6-3	Y	10-2
INSERTB (IB)	6-3	YQMOD (YQM)	5-7
linenumber	4-1	Z	10-2
LISTAB (LT)	11-1	.	10-2
LOCATE (L)	4-1	-	10-2

60455730

CONTROL DATA CORPORATION

XEDIT VERSION 3 REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 1

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(6-2-78)	
Publication No. 60455730	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Front Cover	-	A-1	A						
Command Summary	-	A-2	A						
Title Page	-	B-1	A						
ii	A	B-2	A						
iii/iv	A	B-3	A						
v/vi	A	B-4	A						
vii	A	B-5	A						
viii	A	B-6	A						
1-1	A	B-7	A						
2-1	A	C-1	A						
2-2	A	C-2	A						
2-3	A	C-3	A						
2-4	A	C-4	A						
3-1	A	C-5	A						
3-2	A	C-6	A						
3-3	A	C-7	A						
3-4	A	C-8	A						
4-1	A	C-9	A						
4-2	A	C-10	A						
4-3	A	C-11	A						
4-4	A	C-12	A						
5-1	A	C-13	A						
5-2	A	Index-1	A						
5-3	A	Index-2	A						
5-4	A	Index-3	A						
5-5	A	Comment Sheet	A						
5-6	A	Back Cover	-						
5-7	A								
5-8	A								
6-1	A								
6-2	A								
6-3	A								
6-4	A								
6-5	A								
7-1	A								
7-2	A								
7-3	A								
7-4	A								
8-1	A								
8-2	A								
8-3	A								
8-4	A								
8-5	A								
9-1	A								
9-2	A								
9-3	A								
9-4	A								
10-1	A								
10-2	A								
10-3	A								
11-1	A								
11-2	A								
11-3	A								
12-1	A								
12-2	A								
12-3	A								

PREFACE

XEDIT is an extended interactive text editor developed by the University of Minnesota and available for use the the CONTROL DATA® Network Operating System (NOS). NOS provides network capabilities for time-sharing and transaction processing, in addition to local and remote batch processing, on CDC® CYBER 170 Series, Models 171, 172, 173, 174, and 175 Computer Systems, CDC® CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems, and CDC® 6000 Series Computer Systems.

This manual describes XEDIT use. Each section explains and demonstrates a number of similar XEDIT commands.

This manual assumes that the user is familiar with NOS operation and file concepts. For information on time-sharing processing, the user should refer to the IAF Version 1 Reference Manual or the NOS 1 Time-Sharing User's Reference Manual. For batch processing information, the user should refer to the NOS 1 Batch User's Guide and NOS 1 Reference Manual, volume 1.

RELATED PUBLICATIONS

The following manuals may be of interest to the XEDIT user.

Control Data Publication	Publication Number
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Interactive Facility Version 1 User's Guide	60455260
Network Products Network Terminal User's Instant	60455270
NOS Version 1 Batch User's Guide	60436300
NOS Version 1 Text Editor Reference Manual	60436100
NOS Version 1 Reference Manual, Volume 1	60435400
NOS Version 1 Time-Sharing User's Reference Manual	60435500
NOS Version 1 Time-Sharing User's Guide	60436400
NOS Version 1 Terminal User's Instant	60435800

In the examples in this manual, user input is printed in lowercase letters, and XEDIT output is printed in uppercase letters.

The symbol ⓒⓡ in this manual is used to denote the terminal key that must be pressed in order to send an input line to the system. For most time-sharing terminals, this is the carriage return key. IAF users should refer to the IAF Reference Manual to determine which key applies to them.

DISCLAIMER

This product is intended for use only as described in this document. Control Data Corporation cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLE

INTRODUCTION

XEDIT can be used to manipulate the following types of files.

- Primary files
- Working files
- Indirect access files
- Direct access files

These files can have one or more of the following characteristics.

- Multifile and/or multirecord files (that is, files containing more than one end-of-file or end-of-record mark)
- Files containing either source programs, data, or text (listable files)
- Files prepared in either normal or ASCII mode (the user issues the ASCII command)

The user should not use XEDIT to edit binary files.

XEDIT FEATURES

The following are features of XEDIT.

- Simple command formats
- Multiple commands in a single line
- Verification of user entries; XEDIT automatically lists the file lines modified by a command
- Internal interrupt processing; control is not returned to the operating system
- Editing on the basis of line numbers
- Availability of permanent file commands; a file can be stored without leaving XEDIT
- Easy line modification; strings may be edited on a character-by-character basis
- Editing multifile and/or multirecord files
- Tab control capability
- Window capability; the scope of all string search commands may be restricted to a specific set of columns
- Available for batch processing

Appendix C contains sample editing sessions using these XEDIT features.

CALLING XEDIT

XEDIT can be called from a time-sharing terminal, where editing directives can be given interactively, or in a batch job, where a set of editing directives are processed without further user direction. XEDIT makes certain changes in its processing depending on whether a job is of time-sharing origin or of batch origin.

CALLING XEDIT FOR TIME-SHARING JOBS

To call XEDIT from a time-sharing terminal, the user must complete the following steps. †

1. Log-in to a time-sharing terminal.
2. Access the file to be edited.
3. Select the batch subsystem. ††
4. Enter ATTACH,XEDIT/UN = LIBRARY.
5. Enter the XEDIT control statement.

It is assumed that the user knows the procedure for logging in to his terminal and accessing the file he wants to edit. Direct access files must be attached in write mode so that they can be rewritten with the edited changes. The file to be edited can also be accessed through XEDIT control statement parameters explained later in this section.

Figures 2-1, 2-2, and 2-3 call XEDIT to edit the primary file, a working file, and a direct access file.

```
old,address                    A copy of a permanent indirect access file called address
READY.                         becomes the primary file.
batch                          The batch subsystem is called. XEDIT is attached and
$RFL,0.                        executed following the slash (/) prompt.
/attach,xedit/un=library
/xedit
 XEDIT 3.0
??
```

Figure 2-1. Editing a Primary File

```
get,bfile                      A copy of a permanent indirect access file called bfile becomes a
READY.                         working file. The batch subsystem is called.
batch
$RFL,0.
/attach,xedit/un=library
/xedit,bfile                   XEDIT is called following the slash (/) prompt.
 XEDIT3.0
??
```

Figure 2-2. Editing a Working File

† Users of certain terminals should issue a ROUT command, or if using IAF, a terminal definition command, before calling XEDIT since their terminal carriage return must operate at the correct speed if proper character alignment is to be made. This is necessary for successful execution of the MODIFY, QMOD, and YQMOD commands. Refer to the Time-Sharing User's Reference Manual for a description of the ROUT command and to the IAF Reference Manual for a description of the terminal definition command.

†† Refer to the Time-Sharing User's Reference Manual or the IAF Reference Manual listed in the preface.

```
attach,pascalm/m=w
READY.
batch
$RFL,0.
/attach,xedit/un=library
/xedit,pascalm
 XEDIT 3.0
??
```

A direct access file called pascalm is attached in write mode. The batch subsystem is called.

XEDIT is attached and then called to edit the file pascalm.

Figure 2-3. Editing a Direct Access File

CALLING XEDIT FOR BATCH JOBS

To call XEDIT as part of a batch job, the user includes the statement

ATTACH(XEDIT/UN=LIBRARY)

and the XEDIT control statement in his control statement record. Assuming that the I, L, and FR XEDIT parameters are omitted as described in the following section, XEDIT takes its editing directives from a job deck record following the control statement record and leaves its output on file OUTPUT. The final command in the editing directives record must be END or QUIT; otherwise, XEDIT aborts.

NOTE

If the NEW or OLD control statements are used to create or retrieve the file to be edited, the /ND parameter must be specified on the NEW or OLD control statement so that XEDIT can take its directives from a later record in the job deck. (The NEW and OLD statements return all files including file INPUT if /ND is omitted.)

Figure 2-4 is an example of a batch job that uses XEDIT.

XEDIT CONTROL STATEMENT PARAMETERS

Additional parameters can be specified in the XEDIT control statement. The format used is:

XEDIT(lfn_1, parameter$_1$, parameter$_2$, . . . , parameter$_n$)dcs

A comma can be substituted for the left parenthesis and a period for the right parenthesis. The file name lfn must be the first parameter listed. All other parameters are order-independent.

In brief, the parameters of the XEDIT control statement are:

Parameter	Description
lfn_1	Name of the file to be edited
P	Retrieve and edit permanent file lfn
C	Create a new file lfn_1 (creation mode)
I=lfn_2	Take editing directives from the file specified
I=0	Take all editing directives from the trailing delimited command sequence (following, if the FR parameter is specified, processing of directives from the first line of the file to be edited)
dcs	Trailing delimited command sequence
FR	Take the first editing command from the first line of the file to be edited
NH	Suppress printing of the XEDIT header
L=lfn_3	Put all XEDIT output on the specified file
L=0	No output is generated
B	The job is processed as a batch origin job

The following information describes the effects of specifying or omitting a parameter and examples of how it could be used.

```
JOBNAME.
USER(KS59,3456,FAM1)
CHARGE(4812,RGG3345)
GET(BATCHFL)
ATTACH(XEDIT/UN=LIBRARY)
XEDIT(BATCHFL)
REPLACE(BATCHFL)
7/8/9
P*
C/BOX/P.O.BOX/25
↑P*
END
6/7/8/9
```

The user names the job. He provides user identification and validation information. He accesses an indirect access file called BATCHFL. He attaches and calls XEDIT to edit the file BATCHFL. He replaces the edited file.

The second record of the job deck contains the editing directives to be used.

Figure 2-4. Batch Job Using XEDIT

Parameter	Description
lfn_1	If specified, the file named is rewound, sorted if it is a sequenced primary file, edited as directed, and rewound after processing. If omitted, the primary file is edited. If there is no primary file, the message EMPTY FILE/ CREATION MODE ASSUMED is printed and execution begins under creation mode (refer to explanation of the C parameter). If the file name is omitted, the separators that would appear before and after the file name must be included before other parameters are specified. Example of use: To call XEDIT to edit a local file called ADDRESS: XEDIT(ADDRESS)
P	If specified, file lfn_1 is retrieved from the user's permanent file catalog as a working file. Direct access files are attached in write mode. If the file lfn_1 is not found in the user's permanent file catalog, XEDIT prints an informative message and requests the name of the file to be edited. The user should then enter the file name followed by a C or P parameter, if appropriate. If omitted, the file lfn_1 is assumed to be a local file. Example of use: To call XEDIT to edit a permanent file called ADDR: XEDIT(ADDR, P)
C	If specified, XEDIT begins execution in creation mode. In creation mode, only the following commands are valid. (CR) INSERT TEOR BRIEF LISTAB TOPNULL DEFTAB NOBELLS TRIM DELIMIT RMARGIN VERIFY EXPLAIN STOP WHERE HELP TABS Y INPUT TEOF Z XEDIT leaves creation mode when a line of text exists within the file. Normal editing can follow. Upon exit, a working file is generated with the name, lfn_1, specified in the XEDIT control statement. If a file name lfn_1 is not specified, the file takes the name of the primary file, or if there is no primary file, TAPE1. If omitted, the file to be edited must contain at least one line of text; otherwise, the message EMPTY FILE/CREATION MODE ASSUMED is printed. Example of use: To create a file called ADDR: XEDIT(ADDR, C)
$I=lfn_2$ or I=0	If $I=lfn_2$ is specified, XEDIT takes its editing directives from the specified file after processing commands from the first line of the file to be edited (if the FR parameter is specified) and/or a trailing delimited command sequence. The last command in the specified directives file must be END, QUIT, or STOP; otherwise, XEDIT aborts. If I=0 is specified, XEDIT takes its editing directives from the first line of the file to be edited (if the FR parameter is specified) and from a trailing delimited command sequence. The last command in the delimited command sequence (dcs) must be END, QUIT, or STOP; otherwise, XEDIT aborts. If it is omitted, XEDIT takes editing directives for a batch job from the next record of the job deck. For a time-sharing job, XEDIT takes directives from terminal input or the file INPUT. Examples of use: To call XEDIT to edit a local file called ADDR with directives from a previously created file called INFILE: XEDIT(ADDR, I=INFILE) To call XEDIT to edit a local file called ADDR using only directives from the trailing dcs: XEDIT(ADDR, I=0)PRINT*; END, ADDR, RL
dcs	If specified, XEDIT takes its editing directives from the dcs before taking directives from the input file. Its format is the same as that of a multiple command interactive entry (refer to section 10) except the delimiter character is assumed to be a semicolon unless specified otherwise by the first character of the sequence. The characters which can begin XEDIT commands (letters, numbers, +, /, ↑) and the characters -, *, comma, and space are not recognized as command delimiters.

Parameter	Description
	If omitted, XEDIT takes its directives from the first line of the file to be edited, if the FR parameter is specified, and from the default or specified input file, if I=0 is not specified. Example of use: To call XEDIT to print out the file ADDR before processing the editing directives from the input file B: XEDIT(ADDR, I=B)PRINT*
FR	If specified, XEDIT takes its first editing command from the first line of the file to be edited. The command is assumed to begin with the first nonblank character following two consecutive blanks. XEDIT then takes its commands from the dcs, if one is specified, and then from the input file. If omitted, XEDIT takes its editing directives from the dcs, if one is specified, and then from the input file. Example of use: To call XEDIT to edit a file called ADDR, and to take its first command from the first line of ADDR: XEDIT(ADDR, FR) The following are examples of possible first lines in the file to be edited which would set the delimiter character, tab character, and tab settings. COMPASS: * Y/DELIMIT;/ DEFTAB$/ TABS 10 20 30 FORTRAN: 00100C Y/DELIMIT;/ DEFTAB$/ TABS 10 20 30 MODIFY: */ Y/DELIMIT;/ DEFTAB$/ TABS 10 20 30 BASIC: 00100 REM Y/ DELIMIT;/DEFTAB$/ TABS 10 20 30
NH	If specified, the XEDIT header message (for example, XEDIT 3.0.6) is not printed. If omitted, the header message is printed to verify entry into XEDIT. Example of use: To call XEDIT to edit the primary file without printing the header message: XEDIT(,NH)

L=lfn$_3$ or L=0	If L=lfn$_3$ is specified, all XEDIT output goes on the specified file. The file is not rewound before or after processing. If lfn$_3$ is to be printed on a batch line printer, the file should be shifted by one character position to avoid printer carriage control characters. Refer to the COPYSBF control statement in the NOS Reference Manual, volume 1.
	
	If L=0 is specified, XEDIT generates no output.
	
	If omitted, all XEDIT output goes on the default file OUTPUT.
	
	Examples of use: To call XEDIT to edit the file ADDR and leave its output on file OUTFILE:
	
	XEDIT(ADDR, L=OUTFILE)
	REWIND(OUTFILE)
	COPYSBF(OUTFILE, OUTF)
	REWIND(OUTF)
	ROUTE(OUTF, DC=LP)
	
	To call XEDIT to edit the file ADDR without producing any output:
	
	XEDIT(ADDR, L=0)

B	If specified, XEDIT processes the job as a batch origin job regardless of its method of entry. Any error in the editing directives causes XEDIT to abort. If omitted, jobs are processed according to their origin. Jobs run under the batch subsystem are of time-sharing origin; jobs entered via the SUBMIT control statement are of batch origin. Example of use: To call XEDIT to edit file ADDR if the directives taken from the input file INFILE are correct. If any error exists within the directives, XEDIT aborts, and file ADDR remains in its unedited form. XEDIT(ADDR, I=INFILE, B)

XEDIT CONVENTIONS

All XEDIT commands have the same basic format as explained in General Rules for XEDIT Input. This section also explains how XEDIT provides the means to suppress default conditions, interrupt XEDIT processing without leaving XEDIT, and obtain explanations of commands and error messages during the editing session.

GENERAL RULES FOR XEDIT INPUT

All XEDIT commands follow the same basic pattern in their syntax and parameters and in their required editing data. Knowledge of this basic pattern leads to fewer user errors while editing.

XEDIT COMMAND SYNTAX

XEDIT prints a double question mark (? ?) whenever it expects the user to enter an XEDIT command.† The syntax for an XEDIT command is:

prefixlncommand

The optional prefix characters are:

prefix	Description
/	Advances the pointer one line before processing the command.
^ or ↑	Repositions the pointer to the beginning of the file before processing the prefixed command. This character is octal 76.
X	Temporarily suppresses VERIFY or BRIEF mode while processing the prefixed command.
+	Informs XEDIT that editing data exists on the same line as the command itself when used in conjunction with the DELIMIT, Y, or Z commands; the ADD, INSERT, INSERTB, MODIFY, QMOD, REPLACE, and YQMOD commands are the only commands that use the + prefix. It is ignored when used with other commands. (Refer to section 10.)

Prefix characters can be used in any order and combination and on any legal XEDIT command.

An optional line number prefix (ln) specifies the line number that begins the line at which the prefixed command is to begin processing. If the specified line is not found, the command is not processed, an informative message is printed, and if editing in verify mode, the line at the current pointer position is also printed.

A legal command conforms to the following conventions.

- The command must be spelled correctly.
- Command parameters must follow the sequence shown in that command's description.
- Command parameters must stay within their maximum and minimum numerical limits. When an asterisk is substituted for an n or m parameter, it has a value of 99999.
- The command must be entered at the position where the terminal print element stopped after printing the double question mark.
- No embedded blanks can appear within a command or between a line number prefix and a command.
- The command must be separated from its parameters by either a comma, a space, or for string parameters, the string delimiter.
- Extra spaces may appear between command parameters.

If the user does not enter a valid command after a double question mark prompt (? ?), XEDIT prints an error message and another double question mark prompt (? ?).

XEDIT COMMAND PARAMETERS

In this manual, most XEDIT command formats use the following symbols.

Symbol	Description
n	Number of file lines that the command should process. This number should be a positive integer less than 100000. Its default value is 1. The user can enter an asterisk (*) for n when he wants the command processed until the file pointer reaches the end-of-information. (XEDIT prints an END OF FILE message.)

† A double question mark is not printed if the user has assigned the input or output file to a mass storage device or to a file via the XEDIT control statement parameters I=lfn or I=0.

Symbol	Description
m	Same as n except it specifies the number of occurrences of a string, an end-of-file mark, or an end-of-record mark.
ln	Line number that begins the line to be edited. It must be a positive integer less than 100000.

ENTERING EDITING DATA

XEDIT commands often require that the user enter data indicating how his file should be modified. When editing data is needed, XEDIT prints a single question mark prompt (?). Editing data is requested for ADD, MODIFY, QMOD, YQMOD, and INSERT commands.

Figure 3-1 shows entry of editing data for a MODIFY command.

If the user enters a carriage return after the single question mark (?), XEDIT prints a double question mark (??) indicating that the user can enter his next editing command.

Editing data can also be entered in the same line as the command by using a + prefixed command in a Y or Z command list or by entering a + prefixed command in a delimited command sequence (refer to Submitting Multiple Entries in a Single Line).

STRING DELIMITERS

Certain XEDIT commands allow the user to replace sequences of numbers or characters (strings) that appear within a file line. When a user enters a string command, he separates the string from the command itself, other parameters, and other strings by using a string delimiter. A string delimiter can be any character that is not part of the string. The character must be within the NOS Time-Sharing character set (refer to appendix A) and cannot be a space, number, asterisk, or comma (or a $ under IAF). If the delimiter is alphabetic, a space must separate the command and the delimiter. In this manual, delimiters are represented by a slash (/) character.

For example, if a user wants to locate the next line containing the string IF N = , any of the following LOCATE commands are legal and will accomplish his purpose.

```
LOCATE/IF N = /   (CR)
LOCATE"IF N = "   (CR)
LOCATE ZIF N = Z  (CR)
```

SUPPRESSING DEFAULT CONDITIONS

The XEDIT user may find that he does not want certain aids provided by XEDIT. Therefore, XEDIT has commands to suppress verification, terminal bell ringing, and the printing of file and record marks.

SUPPRESSING VERIFICATION

By default, XEDIT assumes that the user wants verification of the effect of an XEDIT command on his file. For example, suppose a user wants to delete two lines in the file, beginning at the current file pointer position. To do so, the user would issue a DELETE command. XEDIT automatically lists the lines which are deleted so that the user can verify that the command performed the task which was intended. This automatic verification occurs in verify mode.

While verify mode is the default condition for XEDIT, the user can suppress verification if he wants and edit under brief mode. He can select verification or no verification by entering one of the following commands.

Action	Legal Commands
Verification	VERIFY (CR) VERIFY+ (CR) BRIEF- (CR) V (CR)
No Verification	BRIEF (CR) BRIEF+ (CR) VERIFY- (CR) BR (CR)

Figure 3-2 illustrates the difference between verify mode and brief mode. In both instances, the user issues a CHANGE command to delete the string ZIP from every line in the file.

When editing under either verify or brief mode, a user may want to process a single command under the alternate mode without having to issue a BRIEF or VERIFY command. This can be done by prefixing the command with an x, as shown in figure 3-3.

```
?? 110modify
   00110 1544 WILSHIRE ST ZIP 55722
?       2
00110 1244 WILSHIRE ST ZIP 55722
??
```

XEDIT sends a double question mark; user responds by issuing a MODIFY command to alter line 110.
XEDIT prints line 110
XEDIT prints a single question mark; user supplies editing data.
XEDIT prints the modified version of line 110.

Figure 3-1. Entry of Editing Data

Verify Mode

```
?? change/zip//*
00110 1544 WILSHIRE ST  55722
00170 POB 55  55703
00200 8710 14TH ST  55713
END OF FILE
??
```

Brief Mode

```
?? change/zip//*
END OF FILE
??
```

Figure 3-2. Verify Versus Brief Mode

```
?? verify
?? ↑locate/zip/
00110 1544 WILSHIRE ST ZIP CODE 55722
?? xchange/zip code/zip/
??
```

The user issues a VERIFY command.
The user issues a LOCATE command.
XEDIT prints the located line.
The user issues a CHANGE command with an x prefix; XEDIT suppresses its verification.

Figure 3-3. Suppressing Verification

SUPPRESSING PRINTING OF FILE MARKS

XEDIT prints the messages --EOR-- and --EOF-- when it reads an end-of-record mark or end-of-file mark while processing a command. A user may turn off and turn on the printing of these file marks by choosing one of the following commands.

Format	Action
TEOF+ CR	Turns on the printing of --EOF-- messages
TEOF- CR	Turns off the printing of --EOF-- messages
TEOF CR	Toggles printing of --EOF-- messages
TEOR+ CR	Turns on the printing of --EOR-- messages
TEOR- CR	Turns off the printing of --EOR-- messages
TEOR CR	Toggles printing of --EOR-- messages

Figure 3-4 shows suppression of the printing of file marks.

SUPPRESSING TERMINAL BELL RINGING

The NOBELLS command allows the user to suppress bell ringing when XEDIT messages are printed. Its format is:

NOBELLS CR

or

NB CR

Figure 3-5 tests this command.

```
?? print*
*EXECUTIVE MANAGEMENT*
--EOR--
00100 A.B. KRAMER         HQR24
00110 J.J. THOMPSON       SWP19
--EOR--
--EOF--
00120 B.C. MILLER         HQR44
END OF FILE
?? teor
?? teof
?? print*
*EXECUTIVE MANAGEMENT*
00100 A.B. KRAMER         HQR24
00110 J.J. THOMPSON       SWP19
00120 B.C. MILLER         HQR44
END OF FILE
??
```

The user lists a file containing end-of-record and end-of-file marks.

A TEOR command and a TEOF command are issued to suppress the printing of the --EOR-- and --EOF-- messages. The file is listed again.

Figure 3-4. Suppressing Printing of File Marks

```
?? nobells
?? print/
ILLEGAL PARAMETER
?? end
```

The user issues a NOBELLS command.
He then issues a PRINT command with an illegal parameter to test if the bell rings when the error message is printed.

Figure 3-5. Suppressing Terminal Bell Ringing

INTERRUPTING XEDIT PROCESSING

The user can interrupt XEDIT processing without terminating XEDIT execution. To stop the printing of XEDIT output, the user presses the BREAK key (or its equivalent) on non-IAF terminals and the BREAK key (or its equivalent), the colon (:) or the right parentheses [ )], and the CR on IAF terminals. XEDIT prints a double question mark to request entry of the next editing command.

If the interruption occurs while a Z, Y, or delimited command sequence is being processed, the remaining component commands in the command list are skipped.

To stop XEDIT requests for editing data (single question mark prompts), the user presses the carriage return.

ON-LINE REFERENCE AIDS

The user can issue a HELP command for a description of a commandor an EXPLAIN command for an explanation of an XEDIT message.

ASKING FOR ERROR MESSAGE EXPLANATIONS

The EXPLAIN command lets a user request an explanation of an error message that has just been printed. The messages explained by the EXPLAIN command are listed in appendix B. Its format is:

EXPLAIN CR

Figure 3-6 gives the explanation for the error message, NO SUCH COMMAND.

ASKING FOR COMMAND DESCRIPTIONS

The HELP command enables a user to request information about a specific command. If he issues the HELP command without specifying a command or command abbreviation, XEDIT prints a list of all its commands. Its format is:

HELP, command CR

or

H, command CR

Figure 3-7 shows the description of the HELP command.

```
?? get,plan
NO SUCH COMMAND
?? explain

 EXPLAINATION OF-
NO SUCH COMMAND
 THE COMMAND IS ILLEGAL OR AN IMPROPER SEPARATOR
 WAS USED AFTER THE COMMAND.
??
```

The user enters an illegal command. XEDIT prints the appropriate error message. The user requests an explanation. XEDIT prints the explanation of the NO SUCH COMMAND message.

Figure 3-6. Error Message Explanation

```
?? help,print
PRINT $      [P]
=======
Action- Prints $ lines starting at the current pointer  position.  The
pointer is left positioned at the last line printed.
??
```

The user issues a command asking for a description of the PRINT command. XEDIT prints a description.

Figure 3-7. Command Description

POSITIONING THE FILE POINTER

During editing, XEDIT maintains a pointer positioned at the file line currently being processed. The pointer is positioned according to these rules.

Condition	Pointer Position
XEDIT is called.	Beginning-of-information.
A command is processed.	Last line processed by the command (usually the last line printed in verify mode).
Interruption of XEDIT output	Last line processed.
Command processing reaches the end-of-information.	The message END OF FILE is printed, further processing of the command stops, and the pointer is repositioned at the beginning of the file.

POINTER MOVEMENT BY COMMAND PREFIXING

A user can move the pointer before processing an XEDIT command by prefixing the command with either a slash or an up arrow.

/	Advances the pointer one line before processing the prefixed command.
∧ or ↑	Repositions the pointer at the beginning-of-information before processing the prefixed command. (This character is octal 76.)

POINTER MOVEMENT COMMANDS

The XEDIT commands discussed in this section control file pointer positioning for processing of editing commands.

LOCATING LINES VIA LINE NUMBERS

When editing a file sequenced by line numbers, the user can advance the pointer to a line identified by a specific line number by entering the line number after a command prompt. XEDIT begins its search for the specified line number at the current pointer position. The search is circular; the top of the file may be passed in order to position the pointer at the specified line. Processing of the command ends when the pointer is positioned at the line having the specified line number. If no such line exists within the line number sequence, the pointer is positioned at the line with the next highest line number. Its format is:

```
line number  (CR)
```

Figure 4-1 demonstrates this command.

LOCATING LINES VIA SPECIFIED STRINGS

The user can advance the pointer to a line containing a specific string of alphanumeric characters by issuing a LOCATE command. LOCATE commands take three different forms. One form applies when the user wants to find a line containing a specified string. The second form applies when the user wants XEDIT to find a line with the first string followed by a second string. The third form applies when XEDIT is to find a line with the first string not followed by a second string.

When the user issues a LOCATE command with verify mode in effect, XEDIT prints the n lines containing the specified string. The pointer is positioned at the last line printed.

When the user wants to locate a line containing a specific string, the format is:

```
LOCATE/string/n  (CR)
      or
L/string/n  (CR)
```

```
?? 120                     The user issues a line number command to locate line 120.
00120 EXT 6533             XEDIT verifies that the pointer is positioned at line 120.
?? 140                     The user enters line number 140.
00150 O.E. SMITH           XEDIT positions the pointer at line 150 because line 140 does not exist
?? 190                     in this file and 150 is the next highest line number.
190 P.T. BEE               The user enters a 190 to find line 190.
?? 170
00170 POB 55   55703       The user enters a 170 to find line 170.
??                         XEDIT performs a circular search to find line 170 and prints the line.
```

Figure 4-1. Locating Lines Via Line Numbers

string	String of alphanumeric characters which XEDIT attempts to locate. If the user omits the terminating delimiter, XEDIT prints an informative message and processes the command as if a delimiter existed after the last nonblank character.
n	Number of lines to be located containing the specified string. The file pointer is positioned at the last line located. Highest allowable value is 99999; default value is 1. If n is 0, the search is restricted to the current line.

The LOCATE command is used in figure 4-2 to find lines containing a specified string.

When the user wants to locate a line that contains a certain string followed by another string, the format is:

```
LOCATE/string1...string2/n   (CR)

              or

L/string1...string2/n   (CR)
```

string1...string2	string1 followed by string2. If the user omits the terminating delimiter, XEDIT assumes one exists after the last nonblank character.
n	Number of lines to be located containing the specified strings. Highest allowable value is 99999; default value is 1. If n is 0, the search is restricted to the current line.

Figure 4-3 uses the second format of the LOCATE command.

When the user wants to locate a line that contains a certain string not followed by a second string, the format is:

```
LOCATE/string1---string2/n   (CR)

              or

L/string1---string2/n   (CR)
```

string1---string2	string1 not followed by string2. If string1 is not specified, XEDIT searches for n lines not containing string2.
n	Number of lines to be located containing string1 not followed by string2. Highest allowable value is 99999; default value is 1. If n is 0, the search is restricted to the current line.

Figure 4-4 demonstrates the third format of the LOCATE command.

```
?? ^locate/zip/
^0110 1544 WILSHIRE ST ZIP 55722
?? /locate/zip/2
00170 POB 55 ZIP  55
00200 8710 14TH ST ZIP  55713
?? 1/8711/0
STRING NOT FOUND
??
```

The user issues a prefixed LOCATE command to position the pointer at the first line containing the string ZIP.
XEDIT verifies that line 110 is located.
The user issues a slash-prefixed LOCATE command to advance the pointer one line before the search.
XEDIT prints the next two lines containing the string ZIP.

The user issues an abbreviated LOCATE command with a zero count parameter.
The string 8711 is not found on the current line.

Figure 4-2. LOCATE With a Specified String

```
?? ^locate/wilshire...zip 55711/
END OF FILE
?? locate/wilshire...zip 55722/
00110 1544 WILSHIRE ST ZIP 55722
??
```

The user issues a LOCATE command to find a line containing the string WILSHIRE followed by the string ZIP 55711.
XEDIT reads the end-of-information mark without finding the specified string.
The user issues a LOCATE command to find a line containing the string WILSHIRE followed by the string 55722.
XEDIT prints line 110.

Figure 4-3. LOCATE With Two Specified Strings

```
?? ^locate/wilshire/
00010  3780 WILSHIRE AVE ZIP 55722
?? ^locate/wilshire---ave/
00110 1544 WILSHIRE ST ZIP 55722
??
```

The user issues a LOCATE command to find a line containing the string WILSHIRE.
XEDIT prints line 10.
The user issues a LOCATE command to find a line containing the string WILSHIRE but not containing the string AVE.
XEDIT prints line 110.

Figure 4-4. LOCATE With String Not To Be Found

ADVANCING AND REVERSING THE POINTER

When the user wants to advance the pointer toward the end of the file, he enters a NEXT command with the following format.

NEXT n (CR)

or

N n (CR)

n Number of lines that the pointer is advanced. Highest allowable value is 99999; default value is 1.

When the user wants to reverse the pointer toward the beginning-of-information, he enters a NEXT command specifying a negative number of lines as follows:

NEXT -n (CR)

or

N-n (CR)

n Number of lines that the pointer should be moved toward the beginning-of-information. Processing ends when the pointer reaches the beginning-of-information. Maximum value is 99999. If n is 0 or if n is omitted, no pointer movement is performed.

Reverse pointer movements using the NEXT command are much slower than forward pointer movements. Figure 4-5 shows pointer movement in both directions.

POSITIONING POINTER AT TOP OR BOTTOM OF FILE

A user can position the pointer at the beginning-of-information by issuing a TOP command in the following format.

TOP (CR)

or

T (CR)

The user can move the pointer to the bottom of the current record in a file by issuing a BOTTOM command in the following format.

BOTTOM (CR)

or

B (CR)

After positioning the file pointer at the bottom of the current record, the pointer can be moved to the bottom of the next record by entering a NEXT command and another BOTTOM command.

Figure 4-6 uses both the BOTTOM and the TOP commands.

```
?? print 6
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EXTENSION 6533
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXTENSION 227
?? next-3
00120 EXTENSION 6533
?? next 2
00140 166 HASKELL CIRCLE ZIP 55713
??
```

The user issues a PRINT command to list six lines.
XEDIT prints six lines.

The user issues a NEXT command to reverse the pointer three lines.
XEDIT verifies where the pointer is positioned.
The user issues a NEXT command to advance the pointer two lines.
XEDIT verifies where the pointer is positioned.

Figure 4-5. The NEXT Command

```
?? 160
00160 Q.E. SMITH
?? top
?? print
### NAMES/ADDRESSES ARE FICTITIOUS ###
?? bottom
00210 EXT 18
??
```

The user positions the pointer at line 160.
XEDIT prints the location of the pointer.
The user issues a TOP command to position the pointer at the first line in the file; no automatic verification is done.
The user issues a PRINT command.
XEDIT prints the first line in the file.
The user issues a BOTTOM command to move the pointer to the last line in the current record of the file.
XEDIT prints the bottom line of the record.

Figure 4-6. TOP and BOTTOM

POSITIONING POINTER BY LISTING LINES

A user can list lines from a file with the PRINT command. This command begins its listing at the current pointer position. When the listing is finished, the pointer is positioned at the last line printed. PRINT commands have the following format:

PRINT n ⓒⓡ

or

P n ⓒⓡ

n Number of lines which the user wants printed. Maximum value is 99999; default value is 1. If n is *, every line of the file from the current pointer position to the end of information is printed. If n is 0, the current line is printed.

Figure 4-7 demonstrates the PRINT command.

PRINTING THE CURRENT LINE COUNT

The WHERE command gives the user the current line count. Calculated and printed by XEDIT, this is the number of lines between the beginning-of-information and the current pointer position. The format is:

WHERE ⓒⓡ

or

W ⓒⓡ

Figure 4-8 uses the BOTTOM and WHERE commands.

```
?? print
00160 Q.E. SMITH
?? print 4
00160 Q.E. SMITH
00170 POB 55 ZIP  55
190 P.T. BEE
00200 8710 14TH ST ZIP  55713
?? print*
00200 8710 14TH ST ZIP  55713
00210 EXT 18
END OF FILE
?? print*
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EXTENSION 6533
00160 Q.E. SMITH
00170 POB 55 ZIP  55
190 P.T. BEE
00200 8710 14TH ST ZIP  55713
00210 EXT 18
END OF FILE
??
```

The user issues a PRINT command to list the current line. XEDIT prints line 160. The user issues a PRINT command to list four lines. Next, the user lists all lines from the current pointer position to the end-of-information. XEDIT prints lines 190 through 210 and the message END OF FILE. The final PRINT command lists the entire file because the pointer was automatically repositioned to the beginning-of-information by the last command.

Figure 4-7. The PRINT Command

```
?? bottom
ATTACH,XEDI64/UN=XSCB.
?? where
       5
??
```

The user issues a BOTTOM command and a WHERE command to find the number of lines in the record.

Figure 4-8. The WHERE Command

STRING EDITING

The user can modify specific strings of alphanumeric characters within a file line by issuing string editing commands. The following section describes these commands in detail.

ADDING A STRING TO THE END OF A LINE

The user can append a string to the end of one or more file lines with an ADD command. After receiving an ADD command, XEDIT prints a single question mark (?) to prompt entry of the string to be appended. After entry of the string, XEDIT prints the lines to which the entered string has been appended starting at the current pointer position. The ADD command has the following format.

```
ADD n  (CR)

   or

A n  (CR)
```

n	Number of consecutive lines to which the specified string should be appended. Maximum value is 99999; default value is 1; an * can be entered for n when the string is to be appended to every line from the current pointer position to the end of information. When n is 0, the string is appended to the current line.

Figure 5-1 is an example of the use of the ADD command.

REPLACING, DELETING, AND INSERTING STRINGS

CHANGE and CHANGES commands can:

- Replace one string with a different string
- Delete strings from file lines
- Insert strings at the beginning of a line or at the beginning of a window set by the WMARGIN command

This windowing feature can also restrict the columns in which the change can occur. If the user prefers to change strings by visual character-by-character alignment instead of by context, he can use the MODIFY command described later in this section.

REPLACING STRINGS BY CONTEXT

The user can replace one string with a different string of any length and character content with a CHANGE or CHANGES command. However, to use the edited file within NOS, the user must not create a file line longer than 150 characters. The search for the first specified string starts at the line designated by the current pointer position. These formats are:

```
CHANGE/string1/string2/n  (CR)

          or

C/string1/string2/n  (CR)

          or

C/string1a...string1b/string2/n  (CR)

          or

CHANGES/string1/string2/m  (CR)

          or

CS/string1/string2/m  (CR)

          or

CS/string1a...string1b/string2/m  (CR)
```

```
?? 160add
?  jr.
00160 O.E. SMITH JR.
??
```

The user positions the pointer to line 160 and issues an ADD command. XEDIT prints a single ?. The user enters a blank space and then the characters JR.
XEDIT prints the edited line.

Figure 5-1. The ADD Command

string1	String to be replaced.
string1a...string1b	string1a followed by string1b. All characters between and including these two strings are replaced by string2.
string2	String to replace string1 (can be any length).
n	Number of lines containing an occurrence of string1 which should be changed. XEDIT changes every occurrence of string1 in the specified lines. If n is omitted, XEDIT changes the first line found containing string1. Maximum value is 99999. If n is *, XEDIT replaces every occurrence of string1 between the current pointer position and the end of information. If n is 0, the search for string1 is restricted to the current line.
m	Number of occurrences of string1 or string1a... string1b that are to be replaced. If the user omits the m value, XEDIT replaces the next occurrence of the string.

Figure 5-2 shows two uses of the CHANGE command.

DELETING STRINGS BY CONTEXT

CHANGE or CHANGES commands can also be used to delete a string by omitting string2. These commands begin editing at the current line position and continue through n file lines or m occurrences of string1. These formats are:

CHANGE/string//n (CR)

or

C/string1//n (CR)

or

C/string1a...string1b//n (CR)

or

CHANGES/string1//m (CR)

or

CS/string1//m (CR)

or

CS/string1a...string1b//m (CR)

```
?? print* ①
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T.JONES
00110 1244 WILSHIRE ST ZIP 55722
00120 EXT 6533
00160 O.E. SMITH
00170 POB 55 ZIP 55703
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST ZIP 55713
00210 EXT 18
END OF FILE
?? change/ext /853-/* ②
00120 853-6533
00180 853-8837
00210 853-18
END OF FILE
?? 190 ③
00190 P.T. BEE
?? c/bee/bean/
00190 P.T. BEAN
?? 170cs/55/49/ ④
00170 POB 49 ZIP 55703
??
```

① The user issues a PRINT command to list the entire file.

② The user issues a CHANGE command to replace every instance of EXT in the file to 853-. XEDIT verifies which lines are changed.

③ The user locates line 190 and changes the name BEE to BEAN.

④ To change the post office box number without changing the zip code on line 170, the user issues an abbreviated CHANGES command.

Figure 5-2. Use of CHANGE to Modify a Line

string1	String of characters to be deleted.
string1a...string1b	string1a followed by string1b. All characters within and between these two strings in a line are deleted.
n	Number of lines containing string1 that should be processed by this command. Maximum value is 99999; default value is 1. If n is *, every line is deleted from the current pointer position to the end of information. If n is 0, XEDIT does not advance the pointer, and only the current line may be deleted.
m	Number of occurrences of string1 or string1a... string1b to be deleted. If the user omits the m value, XEDIT deletes the next occurrence of the string.

The CHANGE command can delete selected lines as shown in figure 5-3.

INSERTING STRINGS AT THE BEGINNING OF A LINE

A user can insert a string before the first (leftmost) character of a line using a CHANGE or CHANGES command. To do this, he omits string1. The format is:

```
CHANGE//string2/n  (CR)

        or

C//string2/n  (CR)

        or

CHANGES//string2/n  (CR)

        or

CS//string2/n  (CR)
```

```
?? print* ①
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T.JONES
00110 1244 WILSHIRE ST ZIP 55722
00120 EXT 6533
00160 Q.E. SMITH
00170 POB 55 ZIP 55703
00180 EXT 8837
00190 P.TEE BEE
00200 8710 14TH ST ZIP 55713
00210 EXT 18
END OF FILE
?? change/zip//*②
00110 1244 WILSHIRE ST  55722 ③
00170 POB 55  55703
00200 8710 14TH ST  55713
END OF FILE
?? 190changes/ee// ④
00190 P.T BEE
??
```

① The user lists the file.

② The user issues a CHANGE command to delete every occurrence of the string ZIP from every line in the file.

③ XEDIT prints the edited lines.

④ The user then corrects the name in line 190 by issuing a CHANGES command.

Figure 5-3. Selective Deletion via CHANGE Command

string2	String of characters which the user wants to insert before the leftmost character of a line.
n	Number of lines before which string2 is to be inserted. Maximum value is 99999; default value is 1. An * indicates that the insertion should occur before every line from the current pointer position to the end-of-information. If n is 0, XEDIT does not advance the pointer, and the insertion occurs at the current pointer position.

Figure 5-4 shows insertion of a string using the CHANGE command.

RESTRICTING THE STRING SEARCH

The user can restrict the search for a string by issuing one of the following commands.

TRIM (CR)	Ignores trailing blanks on string search commands.
WMARGIN (CR)	Defines column settings which restrict the scope of a string search command when used with the W or A command suffix characters.

IGNORING TRAILING BLANKS

A user can tell XEDIT to ignore trailing blanks on string searches by issuing the TRIM command. The TRIM command has three formats.

TRIM+ (CR)	Ignores trailing blanks.
TRIM- (CR)	Uses trailing blanks.
TRIM (CR)	Toggles the TRIM setting.

The commands whose searches are limited by the TRIM command are:

CHANGE
CHANGES
COPY
COPYD
DELETE
LOCATE
OCTCHANGE

Lines that are entirely blank are not searched when XEDIT ignores trailing blanks. The TRIM command alters searches as shown in figure 5-5.

```
?? print3 ①
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EXT 6533
?? ^change//ex/3 ②
EX00100 M.T. JONES
EX00110 1544 WILSHIRE ST ZIP 55722
EX00120 EXT 6533
??
```

① The user lists three lines.

② The user inserts the string EX before each of the three lines.

Figure 5-4. Use of CHANGE to Insert a String

```
?? print* ①
*EXECUTIVE MANAGEMENT*
EX00100 A.B. KRAMER  HQR24130
EX00110 J.J. JOHNSON SWP19130
EX00130 B.C. MILLER  HQR44130
END OF FILE
?? locate/130 / ②
EX00100 A.B. KRAMER  HQR24130
?? trim ③
?? ^locate/130 /
EX00130 B.C. MILLER  HQR44130④
??
```

① The user lists the file.

② The user attempts to find manager number 130 but instead finds manager number 100 with the facility number HQR24130.
③ The user issues a TRIM command.
④ XEDIT now locates the desired line.

Figure 5-5. The TRIM Command

DEFINING A WINDOW

A user can restrict the scope of a string search to a specified range of columns through use of the WMARGIN command. Once the window columns have been defined, the user can ask for a search restricted to those columns. The user makes that request by appending a W or an A suffix character to the end of any string search command (for example, CHANGEW and LOCATEA). To set the left and right window margins, the WMARGIN command is used as follows:

```
WMARGIN lm rm    (CR)

        or

WM lm rm    (CR)
```

lm Column position setting of the left window margin

rm Column position setting of the right window margin

The left margin must be less than or equal to the right margin.

USING A WINDOW

Once a window has been defined by the WMARGIN command, the user can request that a search be restricted to those columns by appending either a W or A suffix character to the end of any string search command. The legal commands are:

CHANGEW	COPYDW
CHANGEA	COPYDA
CHANGESW	DELETEW
CHANGESA	DELETEA
COPYW	LOCATEW
COPYA	LOCATEA

A suffix character can also be appended to the corresponding command abbreviations.

The W or window suffix character requires that all characters of the string specified by the command reside within the window; otherwise, XEDIT does not find the string.

The A or anchor suffix character requires that the first character of the string specified by the command reside within the window; otherwise, XEDIT does not find the string. All other characters can extend beyond the window.

These suffix characters are used 14 ways in figure 5-6.

The windowing restriction is used to search an employee directory in figure 5-7.

```
?? p ①
ABCDEFGHIJKLM
?? wmargin 4 11②
?? lw/ef...ij/③
ABCDEFGHIJKLM④
?? lw/ef---m/
ABCDEFGHIJKLM
?? lw/cdef...ij/
END OF FILE
?? lw/ef---h/
END OF FILE
?? la/ef...klm/
ABCDEFGHIJKLM
?? la/klm/
ABCDEFGHIJKLM
?? la/ef---m/
END OF FILE
?? la/cdef...ij/
END OF FILE
?? la/def---zz/
ABCDEFGHIJKLM
?? lw/---lm/
ABCDEFGHIJKLM
?? la/---lm/
ABCDEFGHIJKLM
?? cw/ef...ij/xx/
ABCDXXKLM
?? ca/xx/jkl/
ABCDJKLKLM
?? cw//xx/
ABCXXDJKLKLM
??
```

①The user lists his one-line file.

②He sets the window margins to columns 4 and 11 (defghijk inclusive).
③He then tries various commands on this file.
④XEDIT finds the string and prints out the line containing it, or XEDIT searches to the end-of-information without finding the string.

Figure 5-6. Use of the Suffix Characters

```
?? p* ①
DEPT4208 EMP927 KRAMER, SCOTT W.
DEPT4208 EMP208 JOHNSON, JACK J.
DEPT4208 EMP742 MILLER, RICK E.
DEPT4208 EMP698 SCOTT, FRANK R.
END OF FILE
?? wmargin 10 15 ②
?? lw/208/③
DEPT4208 EMP208 JOHNSON, JACK J.④
?? wm 17 17⑤
?? ↑la/scott/ ⑥
DEPT4208 EMP698 SCOTT, FRANK R.⑦
??
```

①The user lists the file.

②He defines the window as columns 10 through 15.

③To locate employee number 208, the user enters an abbreviated LOCATE command with the suffix W so that XEDIT searches only the employee number field.

④XEDIT finds employee number 208, Jack Johnson.

⑤To find the first employee whose last name is Scott, the user redefines the window to be column 17 only.

⑥The user issues an abbreviated LOCATE command with the suffix A.

⑦XEDIT correctly finds the name Frank Scott.

Figure 5-7. Restricting Field of Search

MODIFYING STRINGS CHARACTER-BY-CHARACTER

Three XEDIT commands enable users to alter the contents of a line on a character-by-character basis.† They are MODIFY, QMOD, and YQMOD. These commands differ only in the alignment guides they provide for the modification directives and in the number of lines a single command can modify. The alignment guide for the MODIFY command is the contents of the line at the current pointer position. MODIFY can change only one line at a time. QMOD provides a row of column numbers as an alignment guide for the modification directives and executes those directives on the number of lines specified. YQMOD does not provide an alignment guide but is otherwise identical to QMOD. The procedure for using these commands is:

1. Position the file pointer at the first line to be modified.

2. Issue a modifying command (MODIFY, QMOD, or YQMOD).

3. XEDIT prints the alignment guide, and on the subsequent line, a single question mark.

4. Align and enter the appropriate modification directives under the character or column to be modified (refer to table 5-1).

5. XEDIT prints the modified line or lines.

If no modification directives are entered after the single question mark, no changes are made and no verification is given. The prefixed command XYQMOD suppresses the printing of the alignment guide and the modified lines.

NOTE

The user should not allow a modified line to exceed 150 characters.

The following are the formats of the three modification commands and examples of their use.

MODIFY CR

or

M CR

Figure 5-8 shows the MODIFY command.

QMOD n CR

or

QM n CR

n	Number of lines starting at the current pointer position that is to be modified. Highest allowable value is 99999; default value is 1. If n is *, every line in the file between the current pointer position and the end-of-information is modified. If n is 0, the line at the current pointer position is modified.

QMOD is often used when each line of a file consists of a series of fields each covering a specified range of columns as shown in figure 5-9.

† Since correct alignment is crucial for the successful processing of these commands, the user should be certain that his terminal carriage operates at the correct speed. Adjustment can be made on non-IAF terminals with the ROUT command as described in the NOS Time-Sharing User's Reference Manual and on IAF terminals with the terminal definition command as described in the IAF Reference Manual.

```
?? 10modify ①
  10 XTHIS STRING TO BE MORITFD②
?    &     ^is the #      d # ^ie#③
10  THIS IS THE STRING TO BE MODIFIED③
??
```

① The user issues a MODIFY command with a line number prefix to modify line 10.
② XEDIT lists line 10.
③ The user enters MODIFY directives to alter line 10. XEDIT verifies the changes in line 10.

Figure 5-8. The MODIFY Command

```
?? print*
00100 PART NO= 749322      QUAN=  757     VALUE=  945.50
00110 PART NO= 749323      QUAN= 1298     VALUE=   23.95
00120 PART NO= 749324      QUAN=  446     VALUE=  138.05
00130 PART NO= 749325      QUAN=   15     VALUE= 1650.50
00140 PART NO= 749326      QUAN=  376     VALUE=  182.75
END OF FILE
?? qmod*
  0        1         2         3         4         5         6
  1234567890123456789012345678901234567890123456789012345678901234567890
?                    ^-#                 cost#
00100 PART NO= 7493-22     QUAN=  757     COST=  945.50
00110 PART NO= 7493-23     QUAN= 1298     COST=   23.95
00120 PART NO= 7493-24     QUAN=  446     COST=  138.05
00130 PART NO= 7493-25     QUAN=   15     COST= 1650.50
00140 PART NO= 7493-26     QUAN=  376     COST=  182.75
END OF FILE
??
```

In the first entry, the user issues a PRINT command to list the entire file. XEDIT lists the file. In the next entry, the user issues a QMOD command with n equal to *, indicating that his subsequent modification directives should change every file line. XEDIT prints a sequence of column numbers and a single question mark prompt. The user enters appropriate modification directives in the directives line (a hyphen in front of column 20 and the word COST replacing the word VALUE at column 42). XEDIT verifies the modifications in the file.

Figure 5-9. The QMOD Command

YQMOD n (CR)

or

YQM n (CR)

n Number of lines to be modified by the line of modification directives. Highest allowable value is 99999; default value is 1. If n is *, all lines from the current pointer position to the end of information are modified. If n is 0, the line at the current pointer position is modified.

Figure 5-10 shows a similar editing operation performed with the YQMOD command.

```
 INPUT
?
 EDIT
?? top
?? print*
00100 PART NO= 7493-22     QUAN=  757     COST=  945.50
00110 PART NO= 7493-23     QUAN= 1298     COST=   23.95
00120 PART NO= 7493-24     QUAN=  446     COST=  138.05
00130 PART NO= 7493-25     QUAN=   15     COST= 1650.50
00140 PART NO= 7493-26     QUAN=  376     COST=  182.75
END OF FILE
?? yqmod *
?         ^factory #
00100 FACTORY PART NO= 7493-22     QUAN=  757     COST=  945.50
00110 FACTORY PART NO= 7493-23     QUAN= 1298     COST=   23.95
00120 FACTORY PART NO= 7493-24     QUAN=  446     COST=  138.05
00130 FACTORY PART NO= 7493-25     QUAN=   15     COST= 1650.50
00140 FACTORY PART NO= 7493-26     QUAN=  376     COST=  182.75
END OF FILE
??
```

The user lists the entire file and then enters a YQMOD command specifying that every line in the file is to be modified. XEDIT prints a single question mark. The user counts over seven spaces and inserts the word FACTORY before column 7. XEDIT verifies the modification made in the file.

Figure 5-10. The YQMOD Command

TABLE 5-1. MODIFICATION DIRECTIVES

Directive	Function	Example

↑string#	XEDIT inserts the string of alphanumeric characters between the ↑ and the # before the character pointed to by the ↑.
	
	Issuing a ↑# combination places a # in the file line. If an & or a ↑ is part of the string entered, that special character is treated as a normal character instead of a directive. If a ↑ appears before a string but the terminating # is missing XEDIT assumes the # should appear after the last nonblank character in the directives line.
		`?? 200modify`
		`  00200 8710 14TH ST ZIP 55713`
		`?           ↑ south#`
		`00200 8710 SOUTH 14TH ST ZIP 55713`
		`??`

↑	When a ↑ is entered without any trailing characters, XEDIT inserts a blank in front of the character above the ↑.	`?? 110modify`
		`  00110 1544 WILSHIREST`
		`?                    ^`
		`00110 1544 WILSHIRE ST`
		`??`

#	When a # is entered without a preceding ↑, XEDIT deletes the character above the # and closes up the space left by the deletion.	`?? 170modify`
		`  00170 POB 55 ZIP 55703`
		`?         #`
		`00170 PO 55 ZIP 55703`
		`??`

Blank space	The character above a blank is left unchanged.	`?? 200modify`
		`  00200 8710 SOUTH 14TH ST`
		`?`
		`??`

&	XEDIT deletes the character above an & and replaces it with a blank.	`?? 100modify`
		`  00100 M.T.RJONES`
		`?           &`
		`00100 M.T. JONES`
		`??`

Other alpha-numeric characters	The character in the file line is replaced by the character in the directives line.	`   160`
		`00160 O.E. SMITH`
		`?? 160modify`
		`  00160 O.E. SMITH`
		`?              y  e`
		`00160 Q.E. SMYTHE`
		`??`

LINE EDITING

A user can change entire lines within a file with XEDIT line editing commands. These commands are discussed in the following section.

DELETING LINES

A user can delete entire lines from a file by using any of the following criteria.

- A sequence of lines starting at the current pointer position
- Selective lines on the basis of a specified string
- A single line with a specified string without moving the file pointer if the string is not found
- Selective lines if a specified string occurs within a specified set of columns (refer to the WMARGIN command)
- Selective lines when they are copied to another file (refer to the COPYD command)

The first three listed use the DELETE command. The DELETE command leaves the file pointer positioned at the line after the last deleted line. The formats are:

```
DELETE n  (CR)

   or

D n  (CR)
```

n	Number of lines starting at the current pointer position that is to be deleted. Maximum value is 99999; default value is 1. If n is *, all lines between the current pointer position and the end-of-information are deleted. If n is 0, the line at the current pointer position is deleted.

This format of the DELETE command is used in figure 6-1.

```
DELETE/string/n  (CR)

   or

D/string/n  (CR)
```

string	String of characters that determines if a line should be deleted.
n	Number of lines containing the specified string that is to be deleted. The search starts at the current pointer position. Highest allowable value is 99999; default value is 1. If n is *, every line between the current pointer position and the end-of-information is deleted. If n is 0, the line at the current pointer position is deleted if it contains the specified string.

This second format of the DELETE command is used in figure 6-2.

```
?? 130 ①
00130 B.P. PEEPERS ②
?? delete 3
00130 B.P. PEEPERS
00140 116 WEST ELM DRIVE ZIP 55648
00150 EXT 3222
?? 190 ③
00190 P. T. BEE
?? xdelete*
END OF FILE
??
```

① The user positions the file pointer at line 130 and issues a DELETE command to delete three lines.
② XEDIT verifies the lines deleted.

③ The user next positions the pointer to line 190 and then issues a DELETE command to remove all lines from line 190 to the end-of-information. XEDIT indicates that the end-of-information was encountered but does not verify which lines were deleted because an x prefix was used on the DELETE command.

Figure 6-1. Deleting Selected Consecutive Lines

```
?? delete/ext/*①
00120 EXT 6533②
00150 EXT 5339
00180 EXT 67
00210 EXT 1101
END OF FILE
??
```

①The user issues a DELETE command to delete every line in the file containing the string EXT starting at the current pointer position.
②XEDIT verifies the lines deleted.

Figure 6-2. Deleting Lines Containing a Specified String

DELETE/string1...string2/n CR

or

D/string1...string2/n CR

string1...string2	string1 followed by string2. If XEDIT finds string1 followed by string2 in a line, that line is deleted. Any number of other characters may separate the two strings within the line.
n	Number of lines containing string1 followed by string2 that should be deleted. Maximum value is 99999; default value is 1. If n is *, every line containing string1 followed by string2 from the current pointer position to the end-of-information is deleted. If n is 0, the line at the current pointer position is deleted if it contains string1 not followed by string2.

Figure 6-3 shows line deletion on the basis of two specified strings.

DELETE/string1---string2/n CR

or

D/string1---string2/n CR

string1---string2	string1 is not followed by string2. If XEDIT finds string1 without string2 following it in the same line, that line is deleted.
n	Number of lines containing string1 not followed by string2 that should be deleted. Maximum value is 99999; default value is 1. If n is *, every line containing string1 not followed by string2 from the current pointer position to the end-of-information is deleted. If n is 0, the line at the current pointer position is deleted if it contains string1 followed by string2.

```
?? print* ①
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXT 6533
00130 A.B. MACDONALD
00140 1313 LEMONTREE AVE ZIP CODE 55722
00150 EXT 5339
00160 T.G. SLATER
00170 322 WILSHIRE ST ZIP CODE 55723
00180 EXT 67
00190 R.C. CARTER
00200 6100 WILSHIRE ST ZIP CODE 55722
00210 EXT 1101
END OF FILE
?? delete/wilshire st...55722/* ②
00110 1544 WILSHIRE ST ZIP CODE 55722③
00200 6100 WILSHIRE ST ZIP CODE 55722
END OF FILE
??
```

①The user lists the entire file.

②The user then issues a DELETE command to remove every line in the file that contains the string WILSHIRE ST followed by the string 55722.
③XEDIT verifies which lines are deleted.

Figure 6-3. Deleting Lines Containing Two Specified Strings

REPLACING LINES

The REPLACE command allows the user to replace a specific number of existing file lines starting at the current pointer position with the same number of substitute lines. After the user issues the command, XEDIT prints a single question mark to inform the user that he should enter his first substitute line. XEDIT continues to print single question mark prompts until the specified number of substitute lines has been entered. If the user enters a carriage return immediately after the ? prompt, XEDIT stops processing the REPLACE command and requests the next command. To replace a line with a blank line, the user must enter a space before the carriage return. The format is:

REPLACE n (CR)

or

R n (CR)

n	Number of consecutive lines starting at the current pointer position that is to be replaced. Maximum value is 99999; default value is 1. If n is *, every line from the current pointer position to the end of information is replaced. If n is 0, the line at the current pointer position is replaced.

Figure 6-4 is an example of the use of the REPLACE command.

INSERTING LINES

The user can insert entire new lines into a file, without affecting any existing lines, in three different ways.

- INSERT commands insert a specific number of lines after the current pointer position
- INSERTB commands insert a specific number of lines before the current pointer position
- Entering INPUT mode inserts any number of lines after the current pointer position

INSERT COMMAND

The INSERT command lets a user insert a specific number of lines into a file immediately after the line at the current pointer position. When the user enters the INSERT command, XEDIT prints a single question mark to prompt entry of the first line to be inserted. XEDIT continues to print single question mark prompts until the specified number of lines to be inserted has been entered. If the user enters a carriage return immediately after the ? prompt, XEDIT stops processing the INSERT command and requests the next command. To insert a blank line, the user enters a space before the carriage return. After processing an INSERT command. the file pointer is positioned at the last line inserted. The format is:

INSERT n (CR)

or

I n (CR)

n	Number of lines which the user wants to insert. Maximum value is 99999; default value is 1. If n is 0, XEDIT assumes that one line is to be inserted.

Figure 6-5 demonstrates the INSERT command.

INSERTB COMMAND

When the user wants to insert a specific number of lines into a file before the line at the current pointer position, he issues an INSERTB command which indicates how many lines are to be inserted. Entry of the lines to be inserted is the same as for the INSERT command. The first line entered is the first line of the insertion sequence within the file. After processing an INSERTB command, the pointer is at the same position it occupied before the command was issued. The format is:

INSERTB n (CR)

or

IB n (CR)

n	Number of lines which the user wants to insert. Maximum value is 99999; default value is 1. If n is 0, XEDIT assumes that one line is to be inserted.

Figure 6-6 shows the use of the INSERTB command.

```
?? 170 ①
00170 322 WILSHIRE ST ZIP CODE 55723
?? replace2
? 00170 18 park place apt 111 zip 55704
? 00180 ext 8866 ②
??
```

① The user positions the pointer at line 170 and then issues a REPLACE command to replace line 170 and the line following it.
XEDIT prints a single ?.
The user enters a modified address.
② XEDIT prints a second ? prompt.
The user enters a modified extension number.

Figure 6-4. The REPLACE Command

```
?? 120 ①
00120 EXT 6533
?? insert 3 ②
? 00130 a.b. newton ③
? 00140 166 haskell circle zip 55713④
? 00150 ext 227 ⑤
?? ↑print* ⑥
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXT 6533
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
00160 T.G. SLATER
00170 322 WILSHIRE ST ZIP CODE 55723
00180 EXT 67
00190 R.C. CARTER
00200 6100 WILSHIRE ST ZIP CODE 55722
00210 EXT 1101
END OF FILE
??
```

① The user positions the pointer at line 120.

② He then issues an INSERT command to insert three lines after line 120.

③ XEDIT prints a single question mark and the user enters a line containing an employee name.

④ The user enters a line containing an address.

⑤ The user enters a line containing a phone extension.

⑥ The user lists the entire file to check if the new lines, 130 to 150, were inserted correctly.

Figure 6-5. The INSERT Command

```
?? 160 ①
00160 T.G. SLATER
?? insertb 3②                          ③
? 00157 a.p. macdonald
? 00158 1313 lemontree ave zip 55722
? 00159 ext 5339
?? 140print6④
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
00157 A.P. MACDONALD
00158 1313 LEMONTREE AVE ZIP 55722
00159 EXT 5339
00160 T.G. SLATER
??
```

① The user positions the pointer at line 160.

② He then issues an INSERTB command to insert three lines before line 160.

③ The first line entered is line 157 which is to be the top line of the inserted sequence. The following two lines are inserted.

④ The user lists the sequence to check if the lines were inserted correctly.

Figure 6-6. The INSERTB Command

INPUT MODE

Input mode enables a user to insert an unspecified number of lines following the current file pointer position. To enter input mode, the user performs one of the following.

- Presses carriage return immediately after a ?? prompt
- Enters the INPUT command
- Enters the INPUT command followed by the desired escape character

XEDIT prints the message INPUT, and on the following line, a single question mark. This indicates that the user is in input mode and can enter the first line to be inserted. XEDIT continues to print ? prompts until the user leaves input mode.

The user can leave input mode by entering one of the following after a single question mark prompt.

- A carriage return
- The EDIT command prefixed by the escape character

Batch origin jobs cannot use the carriage return method. They must enter input mode by the INPUT command followed by an escape character and leave input mode by the EDIT command prefixed by the escape character.

If an escape character is specified when entering input mode, the user can process XEDIT commands prefixed by the escape character while in input mode. He is restricted, however, to commands which do not move the file pointer (such as REPLACE instead of DELETE). Whenever the escape character is in the first character position of a line, XEDIT attempts to process the rest of the line as a command. If the command entered after the escape character would move the file pointer, the message COMMAND NOT VALID is printed. If the command specifies a repetition parameter (n or m) other than null or zero, the message ARGUMENT ERROR is printed, and the command is not processed. Processing continues in input mode.

The escape character is recognized as such only when it is in the first column of a line. Therefore, if the tab character is the same as the escape character, the tab character is effective in all columns except the first. If placed there, the rest of the line is interpreted as a command.

The following formats are valid for the INPUT and EDIT commands.

INPUT e (CR)

eEDIT (CR)

e The escape character may be any character except space or the command delimiter set by the DELIMIT command (refer to section 10). When INPUT,, is entered, a comma becomes the escape character.

Figure 6-7 uses the INPUT command to enter input mode.

Figure 6-8 uses (CR) to enter input mode.

```
?? 120 ①
00120 EXT 6533
?? input $ ②
 INPUT ③
? 00121 m.c. greeenway          ⑥
? $change/eee/ee/ ④              \
00121 M.C. GREENWAY ⑤             \
? 00122 867 mission st zip 55744
? $edit⑦
 EDIT
??
```

① The user positions the pointer at line 120.

② The user then issues the INPUT command with a dollar sign ($) as the escape character.
③ XEDIT indicates that it is ready to accept lines to be inserted after the current pointer position.
④ After entering a line, the user notices he misspelled GREENWAY and issues a CHANGE command prefixed by the escape character ($).
⑤ XEDIT verifies that the change was made.
⑥ The user inserts another line of text.
⑦ The user enters the EDIT command prefixed by the escape character to leave input mode.

Figure 6-7. The INPUT Command

```
?? 120 ①
00120 EXT 6533
??
 INPUT②
? 00132 m.c. greenway ③
? 00122 867 mission st zip 55744
? 00123 ext 1500
? 00124 special class**supervisor
? 00125 emp:  12
? 00126 list:  c
? ④
 EDIT
?? t ⑤
?? p*
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXT 6533
00121 M.C. GREENWAY
00122 867 MISSION ST ZIP 55744
00123 EXT 1500
00124 SPECIAL CLASS**SUPERVISOR
00125 EMP:  12
00126 LIST:  C
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
00157 A.P. MACDONALD
00158 1313 LEMON

?? ⑥
```

① The user positions the pointer to line 120.

② The user presses carriage return to enter input mode.
③ The user inserts a series of lines.

④ The user presses carriage return to leave input mode.
⑤ The user issues abbreviated TOP and PRINT commands to list the entire file.

⑥ The user interrupts the listing.

Figure 6-8. The (CR) Command

EDITING LINE NUMBERS 7

The user can issue commands to modify the line numbers that sequence a file. These general conventions apply to all line numbering commands.

- Before any line numbering command is processed, the file pointer is automatically set to the beginning of the file.
- After processing all line numbers within a file and reading the end-of-information mark, XEDIT prints the message END OF FILE to indicate that the specified line number editing is finished. The pointer is automatically repositioned to the beginning of the file.
- A line numbering command must not cause the edited file line to exceed 150 characters if the file is to be processed under NOS.
- No line number can exceed a value of 99999. If a line numbering command would break this rule, the file is restored to its original condition, an informative message is printed, and the pointer is positioned at the beginning of the file.
- Line numbering commands are primarily intended for editing line-numbered card-image text files and programs written in time-sharing FORTRAN Extended. These commands are not practical for use on BASIC programs, since BASIC branch line numbers are not modified by these commands.
- XEDIT does not verify the effects of any line numbering command.

ADDING LINE NUMBERS

Two XEDIT commands allow users to add line numbers to a file.

●	ADDLN	Inserts a line number before each line
●	ADDLNS	Inserts a line number and a blank space before each line

If line numbers already exist in the file, ADDLN and ADDLNS insert an additional set of line numbers.

ADDLN COMMAND

When a user wants to add a line number without a trailing space to each line in a file, he can enter an ADDLN command in the following format.

```
ADDLN ln n  (CR)

     or

ALN ln n  (CR)
```

ln	Line number to be inserted before the first line in a file. No line number can exceed 99999. Default is 1.
n	Number by which each line number in the file is incremented. No line number can exceed a value of 99999. Default is 1.

Figure 7-1 is an example of this command.

ADDLNS COMMAND

When the user wants to add both a line number and a trailing blank space to each line in a file, he enters an ADDLNS command in the following format.

```
ADDLNS ln n  (CR)

     or

ALNS ln n  (CR)
```

ln	Line number to be inserted before the first line in the file. No line number can exceed 99999. Default is 1.
n	Number by which each line number is incremented. No line number can exceed 99999. Default is 1.

Figure 7-2 demonstrates the ADDLNS command.

DELETING LINE NUMBERS

The DELETELN command enables the user to remove every line number from a file. It has no effect on lines without line numbers. The format is:

```
DELETELN  (CR)

     or

DLN  (CR)
```

Figure 7-3 shows a use of this command.

```
?? p* ①
 M.T. JONES
 1544 WILSHIRE ST ZIP CODE 55722
 EXT 6533
 A.B. NEWTON
 166 HASKELL CIRCLE ZIP 55713
 EXT 227
 A.P. MACDONALD
 1313 LEMONTREE AVE ZIP 55722
 EXT 5339
END OF FILE
?? addln 00010 10 ②
END OF FILE
?? p* ③
00010 M.T. JONES
00020 1544 WILSHIRE ST ZIP CODE 55722
00030 EXT 6533
00040 A.B. NEWTON
00050 166 HASKELL CIRCLE ZIP 55713
00060 EXT 227
00070 A.P. MACDONALD
00080 1313 LEMONTREE AVE ZIP 55722
00090 EXT 5339
END OF FILE
??
```

① The user lists a file in which the first character of each file line is a blank.

② He inserts a set of line numbers beginning with 00010 and incrementing by 10.

③ He then lists the file to check the effect of the ADDLN command.

Figure 7-1. The ADDLN Command

```
?? p* ①
Q.E. SMITH
18 PARK PLACE APT 111 ZIP 55704
EXT 8866
P.T. BEE
8710 14TH ST ZIP 55713
EXT 18
END OF FILE
?? addlns 00200 10 ②
END OF FILE
?? p* ③
00200 Q.E. SMITH
00210 18 PARK PLACE APT 111 ZIP 55704
00220 EXT 8866
00230 P.T. BEE
00240 8710 14TH ST ZIP 55713
00250 EXT 18
END OF FILE
??
```

① The user lists a file in which all lines are left-justified.

② He inserts a set of line numbers beginning with 00200 and incremented by 10.

③ He then lists the file to check the effect of the ADDLNS command.

Figure 7-2. The ADDLNS Command

```
?? p* ①
00200 Q.E. SMITH
00210 18 PARK PLACE APT 111 ZIP 55704
00220 EXT 8866
00230 P.T. BEE
00240 8710 14TH ST ZIP 55713
00250 EXT 18
END OF FILE
?? deleteln ②
END OF FILE
?? p* ③
 Q.E. SMITH
 18 PARK PLACE APT 111 ZIP 55704
 EXT 8866
 P.T. BEE
 8710 14TH ST ZIP 55713
 EXT 18
END OF FILE
??
```

① The user lists a file with line numbers.

② He deletes every line number in the file.

③ He then lists the file to check the effect of the DELETELN command.

Figure 7-3. The DELETELN Command

REPLACING LINE NUMBERS

When the user wants to replace an existing set of line numbers with a different set of line numbers, he enters a REPLACELN command in the following format.

REPLACELN ln n (CR)

or

RLN ln n (CR)

ln	Line number to replace the first line number in the file. No line number can exceed 99999. Default is 1.
n	Number by which each line number is incremented. No line number can exceed 99999. Default is 1.

The REPLACELN command shown in figure 7-4 only changes a line if it is preceded by a line number.

FINDING LINES WITHOUT LINE NUMBERS

When a user wants to locate a specific number of lines without line numbers, he enters a FBADL command. This command is a mnemonic for find bad lines, bad lines being lines without line numbers. This command should be used in verify mode so that the lines located are printed. Its format is:

FBADL n (CR)

or

FBL n (CR)

n	Number of lines without line numbers that the user wants to locate. If n is *, XEDIT locates every line in the file not preceded by a line number. Maximum value is 99999; default is 1. The file pointer is positioned at the last line located unless the end-of-information mark is read. If n is 0, XEDIT positions the pointer at the first line located.

Figure 7-5 demonstrates the FBADL command.

```
?? p* ①
00100 Q.E. SMITH
00110 18 PARK PLACE APT 111 ZIP 55704
00120 EXT 8866
00130 P.T. BEE
00140 8710 14TH ST ZIP 55713
00150 EXT 18
END OF FILE
?? replaceln 00500 5 ②
END OF FILE
?? p*③
00500 Q.E. SMITH
00505 18 PARK PLACE APT 111 ZIP 55704
00510 EXT 8866
00515 P.T. BEE
00520 8710 14TH ST ZIP 55713
00525 EXT 18
END OF FILE
??
```

① The user lists a file with line numbers.

② He issues a REPLACELN command specifying that the first line number should be 00500 and the line numbers should be incremented by 5.

③ He lists the file to check the results of the REPLACELN command.

Figure 7-4. The REPLACELN Command

```
?? fbadl* ①
END OF FILE                          ②
?? top
?? fbadl*
### NAMES/ADDRESSES ARE FICTITIOUS ###
END OF FILE
??
```

① The user issues a FBADL command to find all lines without line numbers in the file.

② Because it is in verify mode, XEDIT prints the line found. It also prints the END OF FILE message to indicate that the entire file was searched.

Figure 7-5. The FBADL Command

DELETING LINES WITHOUT LINE NUMBERS

The user can delete lines which do not begin with a line number by entering a DBADL command. Starting at the current pointer position, XEDIT locates and deletes the number of lines without line numbers specified in the command. When XEDIT operates in verify mode, the editor lists each line that is deleted. The format is:

DBADL n ⓒⓡ

or

DBL n ⓒⓡ

n	Number of lines which the user wants deleted starting at the current pointer position. Maximum value is 99999; default value is 1. If n is *, all lines without line numbers between the current pointer position and the end-of-information are deleted. If n is 0, the next line without a line number is deleted.

This command is used in figure 7-6.

```
?? dbadl* ①
### NAMES/ADDRESSES ARE FICTITIOUS ###
END OF FILE②
??
```

① The user issues a DBADL command to delete all lines without line numbers.
② XEDIT prints the line it deleted and the END OF FILE message indicating that the entire file was searched.

Figure 7-6. The DBADL Command

MISCELLANEOUS EDITING COMMANDS

The following XEDIT commands perform various editing functions.

Command	Description
DLBLANKS	Deletes leading blanks from file lines
TOPNULL	Inserts a blank line at the beginning of the file
OCTCHANGE	Converts display code strings
RESTORE	Cancels the effect of the most recent editing commands
DEOF	Deletes end-of-file marks from the file
DEOR	Deletes end-of-record marks from the file
WEOF	Writes end-of-file marks onto the file
WEOR	Writes end-of-record marks onto the file

DELETING LEADING BLANKS

The DLBLANKS command deletes leading blanks from a specified number of lines in the file between the current pointer position and the end-of-information. Blank lines are deleted completely. The format is:

DLBLANKS n Ⓒⓡ

or

DLB n Ⓒⓡ

n	Numer of lines from which leading blanks are to be deleted starting at the current pointer position. Maximum value is 99999; default value is 1. If n is *, all leading blanks between the current pointer position and the end-of-information are deleted. If n is 0, leading blanks are deleted from the current line only.

Figure 8-1 uses the DLBLANKS command.

INSERTING A BLANK LINE AT THE TOP OF A FILE

If the user wants to delete a record mark that begins a file, he must insert a blank line in front of the record mark before he can delete it. This is done using the TOPNULL command. Its format is:

TOPNULL Ⓒⓡ

or

TN Ⓒⓡ

Figure 8-2 shows an example of use of this command.

```
?? p* ①
### NAMES/ADDRESSES ARE FICTITIOUS ###
 Q.E. SMITH
 18 PARK PLACE APT 111 ZIP 55704
 EXT 8866
 P.T. BEE
 8710 14TH ST ZIP 55713
 EXT 18
END OF FILE
?? dlblanks* ②
END OF FILE
?? p*
### NAMES/ADDRESSES ARE FICTITIOUS ###
Q.E. SMITH
18 PARK PLACE APT 111 ZIP 55704
EXT 8866
P.T. BEE
8710 14TH ST ZIP 55713
EXT 18
END OF FILE
??
```

① The user lists the file.

② The user deletes all leading blanks from the file and lists it again for verification.

Figure 8-1. The DLBLANKS Command

```
/xedit,bpgm. ①                ① The user calls XEDIT to edit a file called BPGM.
 XEDIT 3.0 ②                  ② XEDIT prints its header and two end-of-record marks before
--EOR--                          positioning the pointer at the first line in the file.
--EOR--
?? print* ③                   ③ The user lists the file.
00200 PRINT "SQR EXAMPLE"
00210 INPUT N
00220 PRINT SQR(N)
00230 END
END OF FILE
--EOR-- ④                     ④ After reading the end-of-information mark, XEDIT reads the two
--EOR--                          end-of-record marks before positioning the pointer at the first line
?? topnull ⑤                     of the file.
?? xdeor 2 ⑥                  ⑤ The user inserts a blank line at the beginning of the file.
?? ↑delete                    ⑥ He deletes the two record marks without verification, and then
?? print* ⑦                      deletes the blank line he had inserted.
00200 PRINT "SQR EXAMPLE"     ⑦ He lists the file again.
00210 INPUT N
00220 PRINT SQR(N)
00230 END
END OF FILE
??
```

Figure 8-2. The TOPNULL Command

CONVERTING OCTAL STRINGS

The OCTCHANGE command is used to replace one string of octal code (internal display code) with another string of octal code. This enables the user to enter certain terminal line controls (for example, line feeds and carriage returns) into his file that otherwise cannot be specified.

OCTCHANGE oct1 oct2 n (CR)

or

OC oct1 oct2 n (CR)

oct1	Octal display code that represents the existing string to be replaced within the user's file. Each display code character must be represented by an even number of octal digits. Refer to appendix A for a list of valid display codes and their graphic equivalents.
oct2	Octal display code that represents the string that replaces oct1. Each display code character must be represented by an even number of octal digits. Unpredictable results occur when characters are changed to 00 codes. Refer to appendix A for a list of valid display codes and their graphic equivalents.
n	Numer of lines to be processed by the command. Maximum value is 99999; default is 1. If n is *, every line between the current pointer position and the end-of-information is processed. If n is 0, XEDIT does not advance the pointer and processes only the current line.

Figure 8-3 demonstrates the OCTCHANGE command.

CANCELING EDITING OPERATIONS

The user issues a RESTORE command to cancel all editing operations performed since the pointer was positioned at the beginning of the file. This command is entered when the user determines that one or more recently issued commands were incorrectly entered or did not accomplish what the user originally intended. Changes made before the following events cannot be restored.

- An END OF FILE message is printed.
- A TOP command is processed
- An up-arrow (↑) command prefix is processed
- A NEXT -n command is processed
- A find line number (ln) command is processed causing a circular search

The format is:

RESTORE (CR)

or

REST (CR)

Figure 8-4 uses the RESTORE command.

DELETING AND INSERTING RECORDS AND FILE MARKS

The user can both insert and delete record and file marks. DEOR and DEOF commands enable the user to delete these marks, while WEOR and WEOF commands allow the user to insert them.

```
batch
$RFL,0.
/attach,xedit/un=library
/xedit,pass,c ①
 XEDIT 3.0
?? ②
 INPUT
? today's password is:
? orange
?
 EDIT
?? ^print* ③
today's password is:
orange
END OF FILE
?? next
orange
?? xadd ④
? $$wwwwwww$$mmmmmm
?? ^print* ⑤
today's password is:
orange$$wwwwwww$$mmmmmm
END OF FILE
?? octchange 53 7655 ⑥
■■■■■■ ⑦
?? ^print* ⑧
today's password is:
■■■■■■
END OF FILE
??
```

① The user calls XEDIT to create a file called pass.

② He enters input mode and inserts two lines into the file.

③ He lists the file and then positions the pointer at the second line.

④ A string of characters is appended to the end of the line.

⑤ The edited file is printed.

⑥ The user next issues an OCTCHANGE command to replace each dollar sign (octal code 53) with a carriage return (octal code 7655).
⑦ XEDIT verifies the effect of the OCTCHANGE command.
⑧ The user lists the file again.

Figure 8-3. The OCTCHANGE Command

```
?? xlocate/smith/ ①
?? delete 3 ②
00160 Q.E. SMITH ③
00170 322 WILSHIRE ST ZIP CODE 55723
00180 EXT 67
?? restore ④
?? 160 ⑤
00160 Q.E. SMITH
??
```

① The user sets the pointer to the first line containing the string SMITH.
② He then deletes three lines.
③ XEDIT verifies the lines deleted.

④ Seeing that he deleted information on the wrong SMITH, the user issues a RESTORE command.
⑤ To verify the restoration, the user locates line 160.

Figure 8-4. The RESTORE Command

DEOR COMMAND

The DEOR command deletes a specified number of end-of-record marks from the file. Its format is:

DEOR m (CR)

or

DR m (CR)

m	Number of end-of-record marks to be deleted starting at the current pointer position. Maximum value is 99999; default value is 1. If m is *, every end-of-record mark between the current pointer position and the end-of-information is deleted. If m is 0, the next end-of-record mark is deleted.

Figure 8-5 shows use of the DEOR command.

DEOF COMMAND

The DEOF command deletes a specified number of end-of-file marks from the file. Its format is:

DEOF m (CR)

or

DF m (CR)

m	Number of end-of-file marks to be deleted starting at the current pointer position. Maximum value is 99999; default value is 1. If m is *, every end-of-file mark between the current pointer position and the end-of-information is deleted. If m is 0, the next end-of-file mark is deleted.

The DEOF command is used in figure 8-6.

```
?? p* ①

00190 P.T. BEE
00200 8710 14TH ST ZIP 55713
--EOR--
--EOR--
00210 EXT 18
### EXECUTIVE LISTING ###
END OF FILE
?? deor* ②
--EOR--
--EOR--
END OF FILE
??
```

① The user lists the file.

② The user deletes the end-of-record marks. XEDIT verifies that two end-of-record marks were found in the file and deleted.

Figure 8-5. The DEOR Command

```
?? p* ①
00190 P.T. BEE
00200 8710 14TH ST ZIP 55713
--EOR--
--EOF--
210 EXT 18
### EXECUTIVE LISTING ###
END OF FILE
?? deof* ②
--EOR--
--EOF--
END OF FILE
??
```

① The user lists the file.

② He deletes all end-of-file marks in the file.

Figure 8-6. The DEOF Command

NOTE

The DEOF command deletes only end-of-file marks. It does not delete end-of-record marks (such as those created by the WEOF command).

WEOR COMMAND

To insert an end-of-record mark into the file in front of the current pointer position, the user enters a WEOR command in the following format.

WEOR (CR)

or

WR (CR)

Figure 8-7 demonstrates the WEOR command.

WEOF COMMAND

To insert an end-of-file mark into the file in front of the current pointer position, the user enters a WEOF command in the following format.

WEOF (CR)

or

WF (CR)

Figure 8-8 shows the WEOF command.

NOTE

The system inserts an end-of-record mark before the end-of-file mark if one does not already exist.

```
?? locate/executive/ ①
### EXECUTIVE LISTING ###
?? weor ②
?? ↑p*③
00190 P.T. BEE
00210 EXT 18
--EOR--
### EXECUTIVE LISTING ###
END OF FILE
??
```

① The file pointer is positioned at the line containing the word EXECUTIVE.
② The user issues a WEOR command to insert an end-of-record mark before that line.
③ The user lists the file.

Figure 8-7. The WEOR Command

```
?? locate/executive/ ①
### EXECUTIVE LISTING ###
?? weof ②
?? ↑locate/bee/
00190 P.T. BEE
?? p* ③
00190 P.T. BEE
00200 8710 14TH ST ZIP 55713
00210 EXT 18
--EOR--
--EOF--
### EXECUTIVE LISTING ###
END OF FILE
??
```

① The file pointer is positioned at the line containing the word EXECUTIVE.
② The user issues a WEOF command to insert an end-of-file mark before that line.

③ The user lists the file.

Figure 8-8. The WEOF Command

MANIPULATING FILES

The user can manipulate files during editing with the following XEDIT commands.

Command	Description
COPY	Copies a specified set of lines from the edit file onto another file
COPYD	Copies a specified set of lines from the edit file onto another file and deletes those lines from the edit file
READ	Copies the contents of the specified local files onto the edit file
READP	Copies the contents of the specified permanent files onto the edit file
FILE	Saves an intermediate version of the edit file

COPYING LINES ONTO ANOTHER FILE

When the user wants to copy lines from the edit file onto another local file, he can enter either a COPY or a COPYD command. Both commands perform the same function, except that the copied lines are deleted from the edit file when the COPYD command is used. The following rules apply to both COPY and COPYD commands.

- Copying begins at the current pointer position and includes all lines in the file from the current pointer position through the line containing the nth occurrence of the specified string. If XEDIT reads the end-of-information mark while processing these commands, all lines in the file between the initial pointer position and the end-of-information are copied to the local file.
- After the copying is completed, the pointer is positioned at the last line copied or at the beginning of the file if the end-of-information is read.
- When verify mode is in effect, XEDIT prints only those lines containing the specified string.
- Before copying begins, XEDIT rewinds the local file. Later copies onto the same file are added to the end of that file.
- To specify what columns the specified string should occur in, the windowing feature can be used with the COPY and COPYD commands. (Refer to the WMARGIN command.)

The following are the formats of the COPY command and their functions.

Format	Description
COPY fname n Ⓒⓡ	Copies n lines from the edit file onto file fname
COPY fname / string/n Ⓒⓡ	Copies onto file fname all lines from the current pointer position through the nth line containing the specified string or until the end-of-information mark is read.
COPY fname / string1...string2/n Ⓒⓡ	Copies onto file fname all lines from the current pointer position through the nth line containing string1 followed by string2 or until the end-of-information mark is read.
COPY fname / string1---string2/n Ⓒⓡ	Copies onto file fname all lines from the current pointer position through the nth line containing string1 not followed by string2 or until the end-of-information mark is read.
COPY fname / ---string2/n Ⓒⓡ	Copies onto file fname all lines from the current pointer position through the nth line not containing string2 or until the end-of-information mark is read.

fname	Name of the file onto which the specified lines are to be copied; the file name can be from one to seven alphanumeric characters. If fname is OUTPUT, the specified lines are copied to file OUTPUT. At least one blank must appear before and after fname.

n	Number of lines to be copied or the number of lines containing the specified string that are to be located before the copying is completed. Maximum value is 99999; default value is 1. If n is *, all lines from the current pointer position to the end-of-information are copied. If n is 0, a string is specified, and it is the first copy to the specified local file; the file is rewound, the pointer is not advanced, and the current line is copied if it contains the specified string.
string	Specified string of alphanumeric characters to be located.

Figure 9-1 uses the first format of the COPY command.

The COPYD command uses the same parameters as the COPY command. It also is processed the same as the COPY command except that the copied lines are deleted from the edited file. The formats are:

COPYD fname n (CR)

or

COPYD fname /string/n (CR)

or

COPYD fname /string1...string2/n (CR)

or

COPYD fname /string1---string2/n (CR)

or

COPYD fname /---string2/n (CR)

When the user enters the COPYD command with fname specified as NULL, XEDIT deletes the lines without saving them on a local file.

MERGING FILES INTO THE EDITED FILE

To merge one or more files into the edited file, the user enters a READ or READP command. While both commands function the same, READ commands copy local files, and READP commands copy permanent files. The following general rules govern use of both READ and READP commands.

- The specified local or permanent file is copied onto the edited file after the current pointer position, starting with the first record, and continuing until two consecutive end-of-record marks, an end-of-file mark, or an end-of-information mark is detected. Embedded end-of-record marks are maintained.
- Once each copying operation is completed, the pointer is positioned at the last line that was copied onto the edited file.
- XEDIT rewinds the specified files before and after the copying operation.
- If a specified local or permanent file cannot be read (for example, it does not exist or the user entered the wrong file name), XEDIT prints an error message and ends processing of the READ or READP command without copying any other files.

The format of the READP command is:

READP fname$_1$ fname$_2$...fname$_n$ (CR)

fname$_1$ fname$_1$...fname$_n$	Names of the files to be copied onto the edited file in the sequential order in which they are to be copied. A space must separate each name from the preceding name.

This format is used in figure 9-2.

```
?? locate/*division/①
*DIVISION 1*
?? copy div1 /*end/ ②
*END OF DIVISION*
?? end,,rl ③
DIV      REPLACED
DIV      IS A LOCAL FILE
/lnh,f=div1 ④
*DIVISION 1*
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
00157 A.P. MACDONALD
00158 1313 LEMONTREE AVE ZIP 55722
00159 EXT 5339
00160 O.E. SMITH
00170 18 PARK PLACE APT 111 ZIP 55704
00180 EXT 8866
*END OF DIVISION*
/
```

① The user locates the string *DIVISION.

② The user copies all lines from the current pointer position until the next line containing the string *END onto the local file DIV1.

③ The user terminates XEDIT execution.

④ The user lists the local file DIV1.

Figure 9-1. The COPY Command

```
?? p* ①
*EXECUTIVE MANAGEMENT*
00100 A.B. KRAMER  HQR24130
00110 J.J. JOHNSON SWP19130
00120 B.C. MILLER  HQR44120
END OF FILE
?? 120 ②
00120 B.C. MILLER  HQR44120
?? readp divm1 rad1 ③
?? ↑print* ④
*EXECUTIVE MANAGEMENT*
00100 A.B. KRAMER  HQR24130
00110 J.J. JOHNSON SWP19130
00120 B.C. MILLER  HQR44120
*DIVISIONAL MANAGEMENT--PRT DIVISION*
00100 F.R. OLSON            RPT01
00110 M.L. MORRIS           RPT23
00120 B.V. ELLIOT           RPT24
00130 T.C. ROWE             RPT24
00140 H.K. MCGUIRE          RPT89
*RPT RESEARCH AND DEVELOPMENT*
00100 E.R. STANLEY          RPT 55
00110 R.V. HOIM             RPT 56
00120 W.D. ALTHOLZ          RPT 57
00130 K.B. BELLMON          RPT 58
END OF FILE
??
```

① The user lists the file.

② He positions the pointer at line 120.

③ He copies the files DIVM1 and RAD1 onto the edit file.
④ He lists the edit file.

Figure 9-2. The READP Command

The READ command uses the same format and parameters as the READP command. It copies specified local files onto the edit file. To move a group of lines from one part of the file to another, the user can copy the lines onto a local file using the COPY or COPYD commands and then insert the lines at the desired location by positioning the pointer and entering a READ command to copy the local file onto the edit file. The format of the READ command is:

READ fname$_1$ fname$_2$...fname$_n$ (CR)

SAVING AN INTERMEDIATE XEDIT FILE

The FILE command enables the user to save an intermediate version of the edit file. When the FILE command is entered, the following procedure occurs.

1. XEDIT saves a copy of the edit file as specified by the mode parameter.
2. XEDIT prints one or more messages verifying its action.
3. XEDIT requests the next editing command.

The format for the FILE command is:

FILE, fname, mode (CR)

or

F, fname, mode (CR)

fname	Name to be given to the copy of the edit file; it can be from one to seven alphanumeric characters in length. If fname is omitted, the file is given the name given the edit file when XEDIT was called. The commas before and after fname must be included although fname is omitted (for example, FILE, , SAVE).
mode	Type of NOS file to be created. The possible modes are:

Mode	Description
S or SAVE	Saves the edit file as a new indirect access permanent file.
R or REPLACE	Replaces the existing indirect access file fname with the edit file (same as SAVE if no file with that name exists).
L or LOCAL	Makes the edit file a local file; LOCAL mode is illegal for direct access files.
C or COPY	Copies the edit file to file fname. COPY mode is default for direct access files attached in write mode.
RL	Combines the functions of REPLACE and LOCAL modes.
SL	Combines the functions of SAVE and LOCAL modes.

If the mode parameter is omitted, LOCAL mode is assumed for working files and COPY mode for direct access files.

Figure 9-3 demonstrates the FILE command.

```
?? 120 ①          ②
00120 EXT 6533
?? change/6533/6555/
00120 EXT 6555
?? file,address,r ③
ADDRESS REPLACED
??
```

① The user positions the pointer at line 120.

② He changes the extension number.

③ He replaces the indirect access permanent file called ADDRESS with a copy of the edited file.

Figure 9-3. The FILE Command

COMMAND SEQUENCING

The commands described in this section enable the user to repeat a sequence of editing operations at specified points in the file.

SUBMITTING MULTIPLE ENTRIES IN A SINGLE LINE

The user can issue more than one command in a single line by using either of the following methods.

- By specifying with a DELIMIT command the delimiter character to be used to separate a sequence of commands and editing data entered on a single line.
- By using the Z or Y command formats to enter several commands and their editing data on the same line. The command sequence can be repeated using the -n command.

USING DELIMIT COMMAND

The following are basic rules governing use of the DELIMIT command and subsequent delimited sequences.

- Any valid command can appear within the line of delimited commands and editing data.
- The delimiter specified in the DELIMIT command is used to separate commands and editing data.
- The delimit character specified in the DELIMIT command remains in effect until another DELIMIT command is issued.
- The delimiter used in a Y or Z command must not be the same character as the delimiter specified by the last DELIMIT command.
- Both commands and editing data (for example, input related to an ADD command) can appear within a delimited command sequence. To include editing data for a command on the same line as the command, the user should prefix the command with a +. Otherwise, editing data is requested as needed via a single question mark prompt.

DELIMIT commands should be in the following format.

DELIMIT char (CR)

or

DEL char (CR)

char	Character to be used as the delimiter; it must not be alphabetic or a space. To specify a comma as a delimiter, the command DELIMIT, , is entered. If the character is omitted from the command, XEDIT clears the effect of the previous DELIMIT command.

Assuming that the last DELIMIT command specified a semicolon as the delimiter, a delimited command sequence should use the following format.

$\text{cmd}_1;\text{cmd}_2;\ldots;\text{cmd}_n$

cmd	A legal command (or editing data required by a + prefixed command). Each command is processed in order from left to right. The DELIMIT character must not be included in cmd.

If any command is in error, all remaining commands in the list are ignored and the editor requests the next command.

XEDIT expects editing data as the next delimited item following a + prefixed command. Otherwise, editing data is requested as needed via a single question mark prompt.

Figure 10-1 demonstrates the DELIMIT command.

USING Z AND Y COMMANDS

The Z and Y commands can be used to enter a delimited sequence of commands and editing data on a single line. The only difference in the Z and the Y command operation is that for the Z command, XEDIT prints the component commands of the sequence as they are processed, and for the Y command, it does not print them.

The following basic rules apply to both Z and Y commands.

- Any valid XEDIT command except Z and Y commands can be included in the list of component commands. The Y or Z command delimiter should never be the same as the delimiter specified by the last DELIMIT command. If a DELIMIT command is included in a Z or Y command, the DELIMIT character takes effect only after XEDIT has processed all component commands in the Y or Z command sequence.
- If any command in the sequence is in error, all remaining commands in the sequence are ignored, and XEDIT requests the next command.

```
?? print 2 ①
00220 P.B. COLLINS
00230 710 ELM ST                                   ③
?? delimit & ②
?? +xa& zip 55722&+i&00240 ext 726&220p3
00220 P.B. COLLINS
00230 710 ELM ST ZIP 55722
00240 EXT 726
??
```

① The user lists two lines.

② He sets the delimiter character as &.

③ He enters a delimited command sequence containing:

- An abbreviated + and x prefixed ADD command
- The string to be appended by the ADD command
- An abbreviated + prefixed INSERT command
- The line to be inserted
- An abbreviated PRINT command to list three lines beginning at line 220

Figure 10-1. The DELIMIT Command

- Both commands and editing data can appear as part of the command sequence. XEDIT expects editing data as the next delimited item following a + prefixed command.
- The command delimiter can be any alphanumeric or special character not appearing within a component command or editing data item and which is not the delimiter character specified in the last DELIMIT command.

It is suggested that dollar signs or semicolons be used as command delimiters and slashes as string delimiters because those characters are never found in XEDIT commands.

Assuming that $ has been chosen as the command delimiter, Y and Z commands are in the following format.

Z\$cmd$_1$\$cmd$_2$\$...\$cmd$_n$ (CR)

or

Y\$cmd$_1$\$cmd$_2$\$...\$cmd$_n$ (CR)

$	Command delimiter.
cmd	A legal command (or editing data required by a + prefixed command). Each command is processed in order from left to right. The command delimiter must not be included in cmd.

A Z command is used in figure 10-2.

REPEATING COMMANDS

The user can repeat the processing of the preceding command at a different pointer position. To do so, he enters a period followed by the number of lines the pointer should be advanced before the preceding command is processed again. To repeat the processing of a Z or Y command, the user enters a minus sign followed by the number of lines the pointer should be advanced. The formats are:

.n (CR)

or

-n (CR)

.	Process the preceding command (other than Z or Y commands)
-	Process the preceding Z or Y command
n	Number of lines that the pointer should be advanced before processing the preceding command. Default value is 1. If n is 0, the last command is processed without advancing the pointer.

Figure 10-3 shows how to repeat a command.

Figure 10-4 uses a delimited command sequence and the repeat command.

```
?? z$locate/smith/$+add$ jr.$delete/bee/ ①
?? LOCATE/SMITH/ ②
00160 Q.E. SMITH
?? +ADD
00160 Q.E. SMITH JR.
?? DELETE/BEE/
00190 P.T. BEE
??
```

① The user enters a Z command containing three component commands: LOCATE, ADD, and DELETE.
② XEDIT prints each processed command and verification of its action.

Figure 10-2. The Z Command

```
?? print ①
00130 A.B. MACDONALD
?? .3 ②
00157 A.P. MACDONALD
?? .3 ③
00180 EXT 67
??
```

① The user prints a line.

② He enters a .3 to advance the pointer three lines and print that line.

③ He once again advances the pointer three lines and prints the line.

Figure 10-3. Repeating a Command

```
?? del $ ①
?? l/address/$+insert$zip code: ②
ADDRESS:
?? ↑print* ③
NAME:
ADDRESS:
ZIP CODE:
END OF FILE
?? add$.$.$. ④
?   q.e. smith jr. ⑤
NAME:  Q.E. SMITH JR.
?   pob 55
ADDRESS:  POB 55
?   55703
ZIP CODE:  55703
END OF FILE
??
```

① The user sets the delimiter as $.
② He enters a delimited command sequence containing two commands, LOCATE and INSERT.
③ He lists the file.

④ He then enters a delimited command sequence consisting of the ADD command processed four times on consecutive lines.
⑤ For each prompt, he enters an appropriate string and XEDIT prints the appended line.

Figure 10-4. Repeating a Command Sequence

USING TABS AND MARGINS

The commands in this section provide valuable formatting tools for the user. Tab control commands ensure proper spacing of paragraphs, tables, and programs. Margin control commands enable the user to set a right margin, locate lines that extend beyond that margin, and truncate them.

TAB CONTROL

The user can control tab settings during editing by employing the following XEDIT commands.

Command	Description
DEFTAB	Defines a tab character for use in entering editing data with the INSERT, INSERTB, REPLACE, INPUT, and (CR) commands.
TABS	Defines tab stop positions.
LISTAB	Lists the current tab character and tab stop positions.

DEFINING TAB CHARACTER

The user can define a tab character by issuing a DEFTAB command in the following format.

DEFTAB char (CR)

or

DT char (CR)

char	The tab character to be used in entering editing data with the INSERT, INSERTB, REPLACE, INPUT, and (CR) commands. If the user issues a DEFTAB command without specifying a tab character, XEDIT clears the effect of the previous DEFTAB command. There is no default tab character. To specify a comma as the tab character, DEFTAB,, is entered.

Figure 11-1 demonstrates tab character definition and use.

DEFINING TAB POSITIONS

The user can define up to eight tab column positions by issuing a TABS command in the following format.

TABS $t_1t_2t_3 \ldots t_8$ (CR)

or

TAB $t_1t_2t_3 \ldots t_8$ (CR)

t	Tab column positions with increasing values from left to right; default values are 11, 18 and 30. When a TABS command is entered without any t parameters, XEDIT clears the effect of the previous TABS command. Any tab characters included in the input line beyond the last tab stop are copied to the file.

Figure 11-2 shows the setting of tab stops.

LISTING TAB SETTINGS

The user can list the current tab character and tab stop column positions by issuing a LISTAB command in the following format.

LISTAB (CR)

or

LT (CR)

Figure 11-3 uses the LISTAB command.

```
?? deftab $ ①
?? insert
? $mtd$ytd$total ②
?? print ③
          MTD     YTD            TOTAL
??
```

① The user issues a DEFTAB command to establish the dollar sign ($) as the tab character.
② A line is inserted into the file using the tab character to space the column headings according to the default tab column positions, 11, 18, and 30.
③ The user lists the line for verification.

Figure 11-1. Defining a Tab Character

```
?? deftab; ①
?? tabs 5 10 15 20
?? insert 7 ②
? i.  statistics
? ;a.  probability
? ;b.  expected value
? ;;1.  average
? ;;2.  variance
? ;;3.  standard deviation
? ;c.  regression theory
?? ↑l/stat/ ③
I.  STATISTICS
?? p*
I.  STATISTICS
    A.  PROBABILITY
    B.  EXPECTED VALUE
         1.  AVERAGE
         2.  VARIANCE
         3.  STANDARD DEVIATION
    C.  REGRESSION THEORY
END OF FILE
??
```

① The user establishes the semicolon as the tab character and sets the tab column positions at 5, 10, 15, and 20.
② The user inserts seven lines.

③ He then locates the beginning of the group of lines and prints them out as verification.

Figure 11-2. Defining Tab Stops

```
?? deftab; ①
?? tabs 4 8 12
?? listab ②
  ; TABS    4     8     12
??
```

① The user establishes the semicolon as the tab character and sets the tab column positions at 4, 8, and 12.
② He then lists the tab character and settings with the LISTAB command.

Figure 11-3. Listing the Tab Settings

MARGIN AND TRUNCATION CONTROL

The following commands can be used to control the length of file lines.

- RMARGIN Sets the right margin position
- FINDLL Finds lines longer than the current right margin setting
- TRUNCATE Truncates lines longer than the current right margin setting

The user can employ these commands to ensure that no file lines are longer than 150 characters, the maximum length allowed by most NOS line processing routines.

SETTING THE MARGIN

The RMARGIN command sets the column position of the right margin of a file. Lines extending beyond that column can then be located by the FINDLL command and truncated by the TRUNC command. After the RMARGIN command is executed, the pointer retains its original position.

RMARGIN m (CR)

or

RM m (CR)

m The column position of the right margin of the file. It cannot be less than 10.

FINDING LONG LINES

To locate a specific number of long lines, the user issues a FINDLL command. Long lines are lines longer than the current right margin setting (refer to the RMARGIN command). This command should be entered while editing in verify mode so that the located long lines are listed. The format for the FINDLL command is:

FINDLL n (CR)

or

FLL n (CR)

n Number of lines longer than the current RMARGIN setting to be located. If n is *, every long line between the current pointer position and the end of information is found. The maximum value is 99999; default value is 1. The file pointer is located at the last long line found unless the end-of-information is read so that the pointer is positioned at the beginning of the file. If n is 0, XEDIT finds the next long line.

Figure 11-4 uses both the RMARGIN and the FINDLL commands.

TRUNCATING LONG LINES

The TRUNCATE command truncates a specified number of lines at the right margin set by the RMARGIN command. The format is:

```
TRUNCATE n ⓒⓡ

   or

TRUNC n ⓒⓡ
```

n	Number of lines longer than the current RMARGIN setting that the user wants to truncate. If n is *, every long line from the current pointer position to the end-of-information is truncated. Maximum value is 99999; default value is 1. The file pointer is positioned at the last line truncated or at the beginning of file if the end-of-information mark is read. If n is 0, one line is truncated.

Figure 11-5 shows use of this command.

```
?? print* ①                                 ① The user lists the file.
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER          HQR24
EX120 J.J. JOHNSON
EX125 R.E. MILLER          HQR44
EX130 F.R. SCOTT
END OF FILE
?? rmargin 25  ②                            ② He sets the right margin at column 25 and then lists all
?? findll*                                     lines longer than 25 characters.
EX100 S.W. KRAMER          HQR24
EX125 R.E. MILLER          HQR44
END OF FILE
??
```

Figure 11-4. The RMARGIN and FINDLL Commands

```
?? print*  ①                                ① The user lists the file.
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER          HQR24
EX120 J.J. JOHNSON
EX125 R.E. MILLER          HQR44
EX130 F.R. SCOTT
END OF FILE
?? rmargin 25 ②                             ② He sets the right margin at column 25.
?? truncate* ③                                He then truncates all lines in the file longer than
*EXECUTIVE MANAGEMENT*                      ③ 25 characters.
EX100 S.W. KRAMER
EX120 J.J. JOHNSON
EX125 R.E. MILLER
EX130 F.R. SCOTT
END OF FILE
??
```

Figure 11-5. The TRUNCATE Command

The user issues one of the following commands to end his editing session.

• END or QUIT	Terminates editing and saves the edit file
• STOP	Terminates editing and does not save the edit file

TERMINATING XEDIT WITHOUT SAVING EDITING CHANGES

The user issues a STOP command to terminate XEDIT without saving the edited version of the file. The file is left exactly as it appeared before XEDIT was called into execution, assuming that no FILE commands were processed during the session. If the STOP command is issued in batch mode, the first statement after the EXIT control statement is executed; otherwise, the job is terminated.

The STOP command has the following format.

STOP (CR)

Figure 12-1 uses the STOP command.

TERMINATING XEDIT WITH EDITING CHANGES SAVED

When the user wants to terminate XEDIT execution and keep the edited file, he issues a QUIT or END command. Both END and QUIT perform exactly the same function and use identical parameters. The following general rules apply to these commands.

- When the command is entered, the edit file is written onto the specified file fname in accordance with the specified mode parameter. XEDIT then terminates its execution.
- The edit file becomes the primary file only if the filename specified matches the name of the primary file.
- The user should enter SAVE and REPLACE mode parameters with care. SAVE and REPLACE parameters do not write the edit file modifications onto the existing local version of the file. Only permanent files are affected by SAVE and REPLACE parameters.

The END and QUIT commands should be entered in the following format.

END, fname, mode (CR)

or

E, fname, mode (CR)

or

QUIT, fname, mode (CR)

or

Q, fname, mode (CR)

fname	Name (one to seven alphanumeric characters in length) to be given the edit file upon leaving XEDIT. If no file name is entered, XEDIT uses the name specified for the edit file when XEDIT was called. When fname is omitted, the commas before and after fname must be entered.
mode	Type of file the edit file is to become. Valid entries for this parameter are:

```
?? delete 3 ①
00190 R.C. CARTER
00200 6100 WILSHIRE ST ZIP CODE 55722
00210 EXT 1101
END OF FILE
?? restore ②
?? 190
END OF FILE
?? stop ③
 ABORTED
/xedit ④
 XEDIT 3.0
?? 190
00190 R.C. CARTER
??
```

① The user deletes three lines.

② He then attempts to restore the deleted lines but fails to do so because the end-of-information mark was read during the deletion.

③ The user then terminates XEDIT execution to return the file to its original state.

④ When he calls XEDIT again, line 190 is included in the file.

Figure 12-1. The STOP Command

Mode	Description
S or SAVE	Saves the edit file as a new indirect access permanent file
R or REPLACE	Replaces the existing indirect access permanent file fname with the edit file
C or COPY	Rewrites the edit file onto the direct access file of the same name. This mode is default for direct access files attached in write mode
L or LOCAL	Makes the edit file a local file
RL	Combines the functions of REPLACE and LOCAL modes
SL	Combines the functions of SAVE and LOCAL modes

If the mode parameter is omitted, LOCAL mode is assumed for working files, and COPY mode is assumed for direct access files.

Figures 12-2 and 12-3 show use of the END command with REPLACE and RL specified, respectively.

```
?? 120 ①
00120 EXT 6533
?? change/ext/extension/*
00120 EXTENSION 6533
00150 EXTENSION 5339
00180 EXTENSION 67
00210 EXTENSION 1101
END OF FILE
?? end,,r ②
ADDRESS REPLACED
/lnh ③
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXT 6533
00130 A.B. MACDONALD
00140 1313 LEMONTREE AVE ZIP CODE 55722
00150 EXT 5339
00160 T.G. SLATER
00170 322 WILSHIRE ST ZIP CODE 55723
00180 EXT 67
00190 R.C. CARTER
00200 6100 WILSHIRE ST ZIP CODE 55722
00210 EXT 1101
/
```

① The user positions the pointer at line 120. He changes the string EXT to EXTENSION in all lines from line 120 to the end-of-information.

② He then terminates XEDIT and replaces the indirect access file called ADDRESS with the edit file.

③ He then lists the local version of the edit file which does not contain the edited changes.

Figure 12-2. The END Command Using REPLACE Mode

```
?? 120 ①
00120 EXT 6533
?? change/ext/extension/*②
00120 EXTENSION 6533
00150 EXTENSION 5339
00180 EXTENSION 67
00210 EXTENSION 1101
END OF FILE
?? end,,rl ③
ADDRESS REPLACED
ADDRESS IS A LOCAL FILE
/lnh ④
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXTENSION 6533
00130 A.B. MACDONALD
00140 1313 LEMONTREE AVE ZIP CODE 55722
00150 EXTENSION 5339
00160 T.G. SLATER
00170 322 WILSHIRE ST ZIP CODE 55723
00180 EX *INTERRUPTED*
```

① The user positions the pointer at line 120.

② He changes the string EXT to EXTENSION in all lines from 120 to the end-of-information.

③ He then terminates XEDIT and replaces both the indirect access permanent file called ADDRESS and the local file with the edit file.

④ He lists the local file containing the edited changes.

Figure 12-3. The END Command Using RL Mode

NOS TIME-SHARING 64-CHARACTER SET

The character sets for ALGOL and COBOL are listed in their respective reference manuals.

TABLE A-1. NOS 64-CHARACTER SET

CHAR.	CODE (7-BIT OCTAL)	CODE (7-BIT HEXADECIMAL)	INTERNAL DISPLAY CODE (6/12-BIT OCTAL)
:	072	3A	00†
A	101	41	01
B	102	42	02
C	103	43	03
D	104	44	04
E	105	45	05
F	106	46	06
G	107	47	07
H	110	48	10
I	111	49	11
J	112	4A	12
K	113	4B	13
L	114	4C	14
M	115	4D	15
N	116	4E	16
O	117	4F	17
P	120	50	20
Q	121	51	21
R	122	52	22
S	123	53	23
T	124	54	24
U	125	55	25
V	126	56	26
W	127	57	27
X	130	58	30
Y	131	59	31
Z	132	5A	32
0	060	30	33
1	061	31	34
2	062	32	35
3	063	33	36
4	064	34	37
5	065	35	40
6	066	36	41
7	067	37	42
8	070	38	43
9	071	39	44
+	053	2B	45
-	055	2D	46
*	052	2A	47
/	057	2F	50
(	050	28	51
)	051	29	52
$	044	24	53
=	075	3D	54

† IN THE 63-CHARACTER SET, THIS DISPLAY CODE REPRESENTS A NULL CHARACTER. ALSO, USE OF THE COLON IN PROGRAM AND DATA FILES MAY CAUSE PROBLEMS. THIS IS PARTICULARLY TRUE WHEN IT IS USED IN PRINT AND FORMAT STATEMENTS.

CHAR.	CODE (7-BIT OCTAL)	CODE (7-BIT HEXADECIMAL)	INTERNAL DISPLAY CODE (6/12-BIT OCTAL)
(SPACE)	040	20	55
,	054	2C	56
.	056	2E	57
#	043	23	60
[	133	5B	61
]	135	5D	62
%	045	25	63†
"	042	22	64
—	137††	5F	65
!	041	21	66
&	046	26	67
'	047	27	70
?	077	3F	71
<	074	3C	72
>	076	3E	73
@	100	40	74
\	134	5C	75
^	136	5E	76
;	073	3B	77
NULL	—	—	7600
a	141	61	7601
b	142	62	7602
c	143	63	7603
d	144	64	7604
e	145	65	7605
f	146	66	7606
g	147	67	7607
h	150	68	7610
i	151	69	7611
j	152	6A	7612
k	153	6B	7613
l	154	6C	7614
m	155	6D	7615
n	156	6E	7616
o	157	6F	7617
p	160	70	7620
q	161	71	7621
r	162	72	7622
s	163	73	7623
t	164	74	7624
u	165	75	7625
v	166	76	7626
w	167	77	7627
x	170	78	7630
y	171	79	7631

† IN THE 63-CHARACTER SET, THIS DISPLAY CODE REPRESENTS A COLON (:), 7-BIT ASCII CODE 072, 7-BIT HEXADECIMAL CODE 3A.

†† ON TTY MODELS HAVING NO UNDERLINE, THE BACKARROW (←) TAKE ITS PLACE.

TABLE A-1. NOS 64-CHARACTER SET (Contd)

CHAR.	CODE (7-BIT OCTAL)	CODE (7-BIT HEXADECIMAL)	INTERNAL DISPLAY CODE (6/12-BIT OCTAL)
z	172	7A	7632
{	173	7B	7633
|	174	7C	7634
}	175	7D	7635
~	176	7E	7636
DEL	177	7F	7637
NUL	000	00	7640
SOH	001	01	7641
STX	002	02	7642
ETX	003	03	7643
EOT	004	04	7644
ENQ	005	05	7645
ACK	006	06	7646
BELL	007	07	7647
BS	010	08	7650
HT	011	09	7651
LF	012	0A	7652
VT	013	0B	7653
FF	014	0C	7654
CR	015	0D	7655
SO	016	0E	7656
SI	017	0F	7657
DLE	020	10	7660
DC1	021	11	7661
DC2	022	12	7662
DC3	023	13	7663
DC4	024	14	7664
NAK	025	15	7665
SYN	026	16	7666
ETB	027	17	7667
CAN	030	18	7670
EM	031	19	7671
SUB	032	1A	7672
ESC	033	1B	7673
FS	034	1C	7674
GS	035	1D	7675
RS	036	1E	7676
US	037	1F	7677
NULL	—	—	7400
@	100	40	7401
^	136	5E	7402
NULL	—	—	7403
:	072	3A	7404
NULL	—	—	7405
NULL	—	—	7406
`	140	60	7407

MESSAGES

This appendix contains an alphabetical listing of the messages that may appear in a user's output file. Lowercase characters are used to identify variable names or fields. All messages beginning with variable names or characters follow those beginning with A through Z and 0 through 9. These messages are then alphabetized according to the first nonvariable word or character. Messages beginning with any special characters (such as hyphens or asterisks) are alphabetized as if the special character were not present. For example, the message

```
filename SAVED
```

is listed after the messages beginning with A through Z and 0 through 9 and is alphabetized with the messages whose first nonvariable word or character begins with S.

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ABORTED	XEDIT terminated its execution and recorded only those editing changes performed during the editing session which were written to files via FILE, COPY, or COPYD commands.	STOP command has been executed. Recall XEDIT if more editing is to be done.	XEDIT
ARGUMENT ERROR	One of the following: - One string field is needed but is missing. - Two string fields are needed but the second is missing. - While in INPUT mode the user entered a line count or string count parameter value other than zero on a command that might move the pointer, or used the up-arrow or slash (/) prefix characters on a command. - Margin specifications on WMARGIN command are in wrong order. - The Y or Z command list is missing. - In OCTCHANGE an even number of digits was not specified for the octal number.	Reenter last command with correct parameters.	XEDIT
BAD FILE NAME	User did not specify a required file name or specified one that contains bad characters, is more than seven characters in length, or is reserved by XEDIT.	Reenter command with correct file name specified.	XEDIT
BAD TEXT LINE ENCOUNTERED	The last word of a nonempty record does not contain an end of line byte. XEDIT aborts.	Most often user has forgotten to issue a PACK command after leaving TEXT mode. PACK the file and recall XEDIT.	XEDIT
BATCH ABORT-COMMAND ERROR	Command syntax or parameter error (other than DELIMITER) was encountered in batch mode. XEDIT aborts.	Correct error and resubmit job.	XEDIT
CANNOT EDIT EXECUTE ONLY FILES	User attempted to edit a file which is accessed in execute only mode.	Ensure that file name specified is correct. If so, access file in write mode.	XEDIT
COMMAND NOT VALID	User issued an invalid command under INPUT or CREATION mode as follows: - Under INPUT mode, user issued a command which always moves the pointer (for	Enter a valid command such as REPLACE under INPUT mode or INPUT under CREATION mode.	XEDIT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	example DELETE, COPYD, FILE). - Under CREATION mode, user issued a command which assumes the presence of an existing text line (for example, DELETE or INSERTB).		
COMMAND STACKING ERROR	User attempted to recursively call an input source (for example, a Y or Z command is used within another Y or Z command).	Enter legal command.	XEDIT
DELIMITER	Warning message indicating that user omitted closing delimiter of a delimited string parameter. XEDIT executes the command by assuming that a string delimiter appears after the last nonblank character of the string.	Continue editing.	XEDIT
EDIT	User has been returned to EDIT mode.	Enter next editing command.	XEDIT
EMPTY FILE/CREATION MODE ASSUMED	User entered XEDIT without specifying a file to be edited, no primary file is present as default, or the file specified by name or by default is empty. XEDIT automatically enters CREATION mode.	Enter a command valid under CREATION mode.	XEDIT
END OF FILE	XEDIT read the end-of-information mark and terminates execution of the last command. The file pointer is positioned at the beginning-of-information.	Enter next editing command.	XEDIT
END OF INPUT ENCOUNTERED-ABORTED	While in batch mode an end-of-record, end-of-file, or end-of-information mark was encountered on the input file. XEDIT aborts.	Enter an END or QUIT command as the last command in the input file. Resubmit job.	XEDIT
---EOF---	XEDIT read an end-of-file mark. Execution of the command continues until normal termination.	The end-of-file mark can be deleted with the DEOF command and the printing of the message can be turned off by the TEOF command.	XEDIT
---EOR---	XEDIT read an end-of-record mark. Execution of the command continues until normal termination.	The end-of-record mark can be deleted with the DEOR command and the printing of the message can be turned off by the	XEDIT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		TEOR command.	
ERROR IN XEDIT ARGUMENTS	User issued an XEDIT control statement with illegal options specified.	Correct the XEDIT control statement.	XEDIT
FILE FUNCTION ILLEGAL	User is working with direct access files, entered END, FILE, or QUIT command, and specified the LOCAL option or specified the COPY option for a file not attached in WRITE mode.	Change the mode option on the command.	XEDIT
FILE NAME CONFLICT	User issued more than one XEDIT control statement parameter which specified the same file name.	Correct the XEDIT control statement and reenter it.	XEDIT
FILE NOT XEDITABLE	User specified a non-empty file which does not contain a legal line. It cannot be edited by XEDIT.	None.	XEDIT
FR COMMAND STACKING ERROR	XEDIT internal error. The FR XEDIT control statement option has not been processed.	Inform site analyst.	XEDIT
ILLEGAL PARAMETER	User specified one of the following: - An alphabetic DELIMIT command delimiter. - Extra data after the last parameter. - A parameter on a command that does not require a parameter. - An RMARGIN command less than 10 or greater than 160. - Tab stop values not in increasing order or not between 1 and 160.	Correct command and reenter.	XEDIT
INPUT	User has entered INPUT mode.	After the prompting question marks appear, enter lines to be inserted.	XEDIT
LINE NUMBER NOT FOUND, COMMAND NOT EXECUTED	The line number prefix specified a line number not in the file. XEDIT does not execute the command and positions the pointer at the next highest line number.	None.	XEDIT
LINE NUMBER TOO LARGE	A line number formed by the ALN, ALNS, or RLN commands exceeded 99999.	Reenter the command specifying a smaller starting line number or a smaller increment.	XEDIT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
LOCAL FILE ERROR	XEDIT internal error.	Inform site analyst.	XEDIT
NO SUCH COMMAND	User entered an illegal command or an improper separator after the command.	Enter legal command.	XEDIT
NULL LINE-IN CREATION MODE	User issued a PRINT command on an empty file.	Enter a command valid under CREATION mode.	XEDIT
STRING NOT FOUND	The specified string could not be found. The file pointer is not moved.	Extend search for string or continue with next editing command.	XEDIT
TRUNCATION AT n	A processed file line (the nth line in the file) exceeds the maximum line width. XEDIT truncates the line.	None.	XEDIT
XEDIT version	XEDIT header line listing current version.	Enter first editing command.	XEDIT
XEDIT INTERNAL ERROR (ERD). NOTIFY CONSULTANT	An XEDIT internal error has occurred.	Inform site analyst.	XEDIT
YOU DELETED THE ENTIRE FILE	XEDIT cannot locate a line at which to position the pointer; XEDIT aborts.	Recall XEDIT. CREATION mode is assumed until a line of text exists within the file.	XEDIT
0-IN CREATION MODE	User issued a WHERE command; however, no lines exist within the file.	None.	XEDIT
filenam ALREADY PERMANENT	User specified the S or SAVE option on an END, FILE, or QUIT command and the named file is already permanent.	Enter the END, FILE, or QUIT command with an R or REPLACE option.	XEDIT
filenam CANNOT BE ACCESSED	User requested a file which is not local, is an execute only or append only file, or is not available from the user's permanent file area.	Ensure that the file to be edited is local and can be written on.	XEDIT
filenam IS A LOCAL FILE	User specified the L or LOCAL option on an END, FILE, or QUIT command. The edited file is now a local file.	None.	XEDIT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
filenam NOT FOUND NAME EDIT FILE?	User requested a file which could not be found in the user's permanent file area. XEDIT requests the name of a file to edit.	Enter correct name of the file to be edited. If file is permanent, enter ,P after the name. If file is to be created, enter ,C after the name.	XEDIT
filenam REPLACED	User specified the R or REPLACE option on an END, FILE, or QUIT command. The edited file has replaced the permanent file of the same name in the user's permanent file area.	None.	XEDIT
filenam REWRITTEN	An END, FILE, or QUIT command has been executed for a direct access file. The edited file has been copied back to the permanent file of the same name in the user's permanent file area.	None.	XEDIT
filenam SAVED	User specified the S or SAVE option on an END, FILE, or QUIT command. The edited file is now an indirect access permanent file.	None.	XEDIT

Other terminal error messages not listed are issued by NOS. Refer to the Time-Sharing User's Reference Manual or IAF Reference Manual for explanations.

An on-line explanation of any of these messages can be obtained by entering the EXPLAIN command after the message is printed. Refer to the EXPLAIN command for further details.

EXAMPLES OF XEDIT USE

Figures C-1 through C-10 are sample editing sessions which demonstrate a collection of similar XEDIT commands.

Figures C-11 and C-12 are sample batch processing control statement records which use XEDIT to edit a permanent file and to create a file.

```
old,savings
READY.
batch
$RFL,0.
/attach,xedit/un=library  ①
/xedit
 XEDIT 3.0
?? help,print ②
PRINT $      [P]
=======
Action-: Prints $ lines starting at the current pointer  position.  The
pointer is left positioned at the last line printed. ③
?? h tn ④
TOPNULL
=======
Action-: Same as the TOP command except that a blank line  is  inserted
as  the  first  line  of the file. If the edit file has leading record
marks before the first text line, this command can be used to position
the pointer before those marks. ⑤
?? ⑥
```

① User calls XEDIT into execution.

② User issues HELP command; wants information about PRINT command.

③ XEDIT prints information about the PRINT command.

④ User issues an abbreviated HELP command; wants information about the TOPNULL. command.

⑤ XEDIT prints information about TOPNULL.

⑥ XEDIT indicates user should enter another XEDIT command.

Figure C-1. HELP Command

EXAMPLES OF XEDIT USE

Figures C-1 through C-10 are sample editing sessions which demonstrate a collection of similar XEDIT commands.

Figures C-11 and C-12 are sample batch processing control statement records which use XEDIT to edit a permanent file and to create a file.

```
 old,savings
 READY.
 batch
 $RFL,0.
 /attach,xedit/un=library  ①
 /xedit
  XEDIT 3.0
?? help,print ②
PRINT $      [P]
=======
Action-: Prints $ lines starting at the current pointer  position.  The
pointer is left positioned at the last line printed. ③
?? h tn ④
TOPNULL
=======
Action-: Same as the TOP command except that a blank line  is  inserted
as  the  first  line  of the file. If the edit file has leading record
marks before the first text line, this command can be used to position
the pointer before those marks. ⑤
?? ⑥
```

① User calls XEDIT into execution.

② User issues HELP command; wants information about PRINT command.

③ XEDIT prints information about the PRINT command.

④ User issues an abbreviated HELP command; wants information about the TOPNULL. command.

⑤ XEDIT prints information about TOPNULL.

⑥ XEDIT indicates user should enter another XEDIT command.

Figure C-1. HELP Command

```
old,savings
READY.
batch
$RFL,0.                                      ①
/attach,xedit/un=library
/xedit
 XEDIT 3.0
?? print* ②
00100 PRINT TAB(10),"DAYDREAM SAVINGS PLAN"
00110 PRINT
00120 PRINT "DAYS","DAILY AMOUNT","TOTAL SAVINGS"
00130 LET N=1
00140 LET D=.01
00150 LET A=0
00160 LET A=A+D
00170 PRINT USING 00180,N,D,A
00180 ^  ##          #########.##     #########.##   ③
00190 LET D=2+D
00200 LET N=N+1
00210 IF N<31 THEN 00160
00220 END
END OF FILE
?? print  ④
00100 PRINT TAB(10),"DAYDREAM SAVINGS PLAN" ⑤
?? 170 ⑥
00170 PRINT USING 00180,N,D,A
?? x200 ⑦
?? next    ⑧
00210 IF N<31 THEN 00160
?? ↑next3 ⑨
00130 LET N=1
?? next -3  ⑩
00100 PRINT TAB(10),"DAYDREAM SAVINGS PLAN"
?? locate/let...+/* ⑪
00160 LET A=A+D
00190 LET D=2+D
00200 LET N=N+1
END OF FILE
?? locate/let/3 ⑫
00130 LET N=1
00140 LET D=.01
00150 LET A=0
?? print ⑬
00150 LET A=0
?? bottom ⑭
00220 END
??
```

① User calls XEDIT.

② User issues PRINT command to list entire contents of the file (that is, * is entered as the n parameter).

③ XEDIT lists entire file.

④ User issues PRINT command; by default since no n parameter appears, XEDIT assumes only 1 line should be listed.

⑤ XEDIT prints the line indicated by the current pointer position.

NOTE

Since the first PRINT command reached an end-of-file, the pointer was repositioned to the top of the file.

⑥ User enters a particular line number (170) to place pointer at that position. By default, verify mode is in effect.

⑦ User enters line number 200 but indicates that XEDIT should not verify the current pointer position (that is, an X prefixes the user entry).

⑧ User issues NEXT command to advance the pointer by one line; XEDIT verifies the pointer position.

⑨ User issues an up-arrow (↑) prefix to reposition pointer to the beginning of the file and to NEXT command. Advances the pointer by three lines.

⑩ User issues NEXT command to move the pointer backwards by three lines.

⑪ User issues LOCATE command to find the third line (from the current pointer position) containing the string LET.

⑫ User issues LOCATE command to find all lines in the file containing a combination of two strings (LET and +).

⑬ User issues PRINT command; pointer is positioned at the beginning of the file.

⑭ User issues BOTTOM command; pointer is now positioned at the last line in the file.

Figure C-2. Pointer Movement Commands

```
old,savings
READY.
batch
$RFL,0.
/attach,xedit/un=library                              ①
/xedit
 XEDIT 3.0
?? print*  ②
00100 PRINT TAB(10),"DATEDREAM SAVING PLAN"
00110 PRINT
00120 PRINT "DAYS","DAILY AMOUNT","TOTAL SAVINGS"
00130 LET N=1
00140 LET D=.01
00150 LET A=0
00160 LET A=A+D
00170 PRINT USING 00180,N,D,A
00180 ^   ##          #########.##     #########.##  ③
00190 LET D=2+D
00200 LET N=N+1
00210 IF N<31 THEN 00160
00220 END
END OF FILE
?? locate/tab/  ④
00100 PRINT TAB(10),"DATEDREAM SAVING PLAN"
?? change/(10)/(13)/ ⑤
00100 PRINT TAB(13),"DATEDREAM SAVING PLAN"
?? change/</<=/  ⑥
00210 IF N<=31 THEN 00160
?? ↑c/let//*  ⑦
00130  N=1
00140  D=.01
00150  A=0
00160  A=A+D      ⑧
00190  D=2+D
00200  N=N+1
END OF FILE
?? locate/n=/ ⑨
00130  N=1
?? add  ⑩
? .75   ⑪
00130  N=1.75  ⑫
?? 100modify ⑬
  00100 PRINT TAB(13),"DATEDREAM SAVING PLAN" ⑭
?                        y#            ^s # ⑮
00100 PRINT TAB(13),"DAYDREAM SAVINGS  PLAN"  ⑯
??
```

① User calls XEDIT to edit a file called SAVINGS.

② User issues PRINT command to list entire file.

③ XEDIT lists the SAVINGS file.

④ User issues LOCATE command to move pointer to first line containing the string TAB; XEDIT verifies which line was located.

⑤ User issues CHANGE command to replace each reference to (10) with (13) in the first line containing a (10); XEDIT verifies how the change was made.

⑥ User issues CHANGE command to replace each reference to < with < = in the first line containing a < sign; XEDIT verifies how the change was made.

⑦ User issues an abbreviated CHANGE command with an up-arrow (↑) prefix to delete all references to LET with a blank (in other words, delete the LET references) throughout the entire file.

⑧ XEDIT indicates which lines were affected by the preceding CHANGE command.

⑨ User issues LOCATE command to find the first line containing an N=string; XEDIT places pointer to line 130.

⑩ User issues ADD command to attach information onto the end of the line designated by the current pointer position (that is, line 130).

⑪ User specifies that .75 should be added to the end of line 130.

⑫ XEDIT verifies the change which resulted because of the above ADD command.

⑬ User issues MODIFY command to alter line 100.

⑭ XEDIT prints line 100.

⑮ User issues MODIFY directives to indicate how line 100 should be altered.

⑯ XEDIT verifies how line 100 was affected by the above MODIFY command.

Figure C-3. String Editing Commands

```
old,depart
READY.
batch
$RFL,0.                                         ①
/attach,xedit/un=library
/xedit
 XEDIT 3.0
?? print*
*DEPARTMENT 4208*
D4208 E927 EX100 KRAMER, SCOTT W.   HQR24-130
D4208 E208 EX130 JOHNSON, JACK J. SWP19-250
D4208 E442 EX120 MILLER, RICK HQR44-140           ②
DR208 E698 EX125 SCOTT, FRANK R.  MXJ97-220
END OF FILE
?? locate/130
DELIMITER                                         ③
D4208 E927 EX100 KRAMER, SCOTT W.   HQR24-130
?? trim ④
?? ↑locate/130 /
D4208 E208 EX130 JOHNSON, JACK J. SWP19-250
?? wmargin 7 10 ⑤
?? ↑lw/208/ ⑥
D4208 E208 EX130 JOHNSON, JACK J. SWP19-250
?? wm 18 18 ⑦
?? ↑la/scott/ ⑧
DR208 E698 EX125 SCOTT, FRANK R.  MXJ97-220
??
```

① User calls XEDIT to edit a file called DEPART.

② User issues PRINT command to list entire file; XEDIT lists file.

③ User attempts to find executive number 130 but instead finds executive 100 since his facility number HQR24-130 happened to also have the number 130 in it.

④ To correctly locate executive 130, the user issues a TRIM command to tell XEDIT to ignore the trailing spaces (blanks) to the right of all facility numbers such as HQR24-130.

⑤ The user intends to search for employee number 208, so the WMARGIN command is issued to restrict the search window to just the employee number field (columns 7 through 10 inclusively).

⑥ Once the window has been defined, the user issues an abbreviated LOCATE command (the window W postfix) to specify that XEDIT should look at only the employee number field during the search. XEDIT responds by finding employee Jack Johnson number E208 as requested, instead of the first line with D4208 in it.

⑦ Next the user wants to find the first employee whose last name is Scott. Since the last name field begins in column 18, the user issues an abbreviated WMARGIN command to redefine the window to be just that column.

⑧ The user finally issues an abbreviated LOCATE command with the anchor postfix A. The up-arrow (↑) prefix tells XEDIT to begin the search at the top of the file. XEDIT correctly finds Frank Scott instead of Scott Kramer.

Figure C-4. String Search Control Commands

```
 old,plan                                  }
 READY.                                    }
 batch                                     }
 $RFL,0.                                   } (1)
 /attach,xedit/un=library                  }
 /xedit                                    }
  XEDIT 3.0                                }
?? print*  (2)
TABLE ONE
TABLE TWO  (3)
END OF FILE
?? deftab ; (4)
?? tabs 20 30 40 50 60  (5)
?? listab (6)
  ; TABS    20     30     40     50     60
?? (7)
 INPUT
? ;;profit picture                                                 }
? ;;==============                                                 }
? ;1q;2q;3q;4q;total                                               }
? ;--;--;--;--;-----                                               }
? sales revenue;48,578;51,027;53,598;56,300;$209,503               } (8)
? total expenses;42,273;43,952;45,709;47,548;$179,482              }
? operating profit;6,305;7,075;7,889;8,752;$30,021                 }
? ----------------                                                 }
? (9)
 EDIT
??-next (10)
TABLE TWO
?? tats 24 34 44  (11)
?? listab (12)
  ; TABS    24     34     44              60
?? (13)
 INPUT
? ;;cash flow forecast                                  }
? ;;==================                                  }
? ;july;aug;sept                                        }
? ;----;---;----                                        }
? total cash available;450,000;452,300;459,316          } (14)
? total cash required;378,100;388,600;397,908           }
? ending cash balance;71,900;63,700;61,408              }
? -----------------                                     }
? (15)
 EDIT
?? ^print*  (16)
TABLE ONE
                              PROFIT PICTURE                          }
                              ==============                          }
                  1Q          2Q          3Q          4Q        TOTAL }
                  --          --          --          --        ----- }
SALES REVENUE     48,578      51,027      53,598      56,300    $209,503
TOTAL EXPENSES    42,273      43,952      45,709      47,548    $179,482
OPERATING PROFIT  6,305       7,075       7,889       8,752     $30,021 }
----------------                                                      }
TABLE TWO                                                             } (17)
                                  CASH FLOW FORECAST                  }
                                  ==================                  }
                      JULY        AUG         SEPT                    }
                      ----        ---         ----                    }
TOTAL CASH AVAILABLE  450,000     452,300     459,316                 }
TOTAL CASH REQUIRED   378,100     388,600     397,908                 }
ENDING CASH BALANCE   71,900      63,700      61,408                  }
-------------------                                                   }
END OF FILE
??
```

(1) User calls XEDIT to edit a file called PLAN.

(2) User issues PRINT command to list entire file.

(3) XEDIT lists file.

(4) User issues DEFTAB command to establish a semicolon(;) as the tab character.

(5) User issues TABS command to establish tab stop column positions 20, 30, 40, 50 and 60.

(6) User issues LISTAB command to list the current tab character and tab stop column positions.

(7) User presses carriage return to insert an unknown number of lines after the TABLE ONE line.

(8) User enters several new lines with the tab character used where tabbing is desired.

(9) User presses carriage return to terminate the insertion of new lines.

(10) User issues NEXT command to advance the pointer by one line; XEDIT verifies the pointer position.

(11) User issues another TABS command to reset the tab stops to positions 24, 34, and 44.

(12) User issues LISTAB command to list the current tab character and tab stop column positions.

(13) User presses carriage return to insert an unknown number of lines after TABLE TWO line.

(14) User adds several new lines with the tab character used where tabbing is desired.

(15) User presses carriage return to terminate the insertion of new lines.

(16) User issues a PRINT command with an up-arrow (↑) prefix to list the entire file.

(17) XEDIT lists the file verifying that tabbing was correctly performed.

Figure C-5. Tab Control Commands

```
old,manager
READY.
batch
$RFL,0.                                ①
/attach,xedit/un=library
/xedit
 XEDIT 3.0
?? print* ②
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER      HQR24-130
EX110 J.J. JOHNSON     SWP19-130  ③
EX120 R.E. MILLER      HQR44-130
EX125 F.R. SCOTT       MXJ97
END OF FILE
?? rmargin 30 ④
?? findll*
EX100 S.W. KRAMER      HQR24-130
EX110 J.J. JOHNSON     SWP19-130  ⑤
EX120 R.E. MILLER      HQR44-130
END OF FILE
?? trunc*
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER      HQR24-13
EX110 J.J. JOHNSON     SWP19-13   ⑥
EX120 R.E. MILLER      HQR44-13
EX125 F.R. SCOTT       MXJ97
END OF FILE
??
```

① User calls XEDIT to edit a file called MANAGER.

② User issues PRINT command to list entire file.

③ XEDIT lists file.

④ User sets the right margin to column 30.

⑤ User requests XEDIT to find all lines that are larger than 30 characters (current RMARGIN setting) by issuing the FINDLL command; XEDIT responds accordingly.

⑥ User requests XEDIT to truncate all lines larger than 30 characters (current RMARGIN setting) by issuing the TRUNC command; XEDIT responds accordingly.

Figure C-6. Margin Control Commands

```
old,add5                          }
READY.                            }
batch                             }
$RFL,0.                           } ①
/attach,xedit/un=library          }
/xedit                            }
 XEDIT 3.0                        }
?? print* ②
### NAMES/ADDRESSES ARE FICTITIOUS ###  }
00100 M.T. JONES                        }
00110 1544 WILSHIRE ST ZIP 55722        }
00120 EX6533                            }
00160 Q.E. SMITH                        }
00165 APT 23                            } ③
00170 POB 55 ZIP 99107                  }
00180 EXT 8837                          }
00190 P.T. BEE                          }
00200 8710 14TH ST                      }
00210 EXT 18                            }
END OF FILE                             }
?? 190delete3 ④
00190 P.T. BEE        }
00200 8710 14TH ST    } ⑤
00210 EXT 18          }
END OF FILE
?? locate/pob ⑥
DELIMITER
00170 POB 55 ZIP 99107
?? replace ⑦
? 00170 138 chess ave ⑧
?? insertb ⑨
? 00166 c/0 b.t. smith sr ⑩
?? bottom ⑪
00180 EXT 8837
?? ⑫
 INPUT
? 00190 t.r. doesinger }
? 00200 662 elm st     } ⑬
? 00210 ext 7714       }
? ⑭
 EDIT                                    ⑯
?? ↑insert ⑮
? #sample line editing command session#
?? ↑print* ⑰
### NAMES/ADDRESSES AE FICTITIOUS ###  }
#SAMPLE LINE EDITING COMMAND SESSION#  }
00100 M.T. JONES                       }
00110 1544 WILSHIRE ST ZIP 55722       }
00120 EX6533                           }
00160 Q.E. SMITH                       }
00165 APT 23                           }
00166 C/O B.T. SMITH SR                } ⑱
00170 138 CHESS AVE                    }
00180 EXT 8837                         }
00190 T.R. DOESINGER                   }
00200 662 ELM ST                       }
00210 EXT 7714                         }
END OF FILE                            }
??
```

① User calls XEDIT to edit a file called ADD5.

② User issues PRINT command to list the entire file.

③ XEDIT lists ADD5 file.

④ User issues a DELETE command with a 190 prefix to specify that he wants to position pointer to line 190 and wants to delete the next three file lines starting at line 190.

⑤ XEDIT verifies which three lines were deleted.

⑥ User issues LOCATE command to find first line containing the string POB; XEDIT indicates line 170 is found in response.

⑦ User issues REPLACE command to insert an entirely new line of information in place of line 170.

⑧ User indicates what new information should replace line 170.

⑨ User issues INSERTB command to enter an entire line of information in front of the current pointer position (that is, in front of line 170).

⑩ User enters the new information that should be placed in the line which will precede line 170.

⑪ User issues BOTTOM command.

⑫ User presses carriage return to insert an unknown number of lines after line 180.

⑬ User enters three new lines.

⑭ User presses carriage return to terminate the insertion of new lines.

⑮ User issues INSERT command with an up-arrow (↑) prefix character to enter one new line after the first line in the file (the pointer is positioned to the first line in the file).

⑯ User enters the new line.

⑰ User enters a PRINT command with an up-arrow (↑) prefix character to list the file in its edited form.

⑱ XEDIT lists the edited file.

Figure C-7. Line Editing Commands

```
old,add5
READY.
batch
$RFL,0.                                  ①
/attach,xedit/un=library
/xedit
 XEDIT 3.0
?? print* ②
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EX6533
00160 Q.E. SMITH
00165 APT 23
00170 POB 55 ZIP 99107                    ③
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST
00210 EXT 18
END OF FILE
?? brief ④
?? change/zip/zip code/* ⑤
END OF FILE
?? change/ext/extension/* ⑥
END OF FILE
?? print* ⑦
### NAMES/ADDRESSES AE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EX6533
00160 Q.E. SMITH
00165 APT 23                             ⑧
00170 POB 55 ZIP CODE 99107
00180 EXTENSION 8837
00190 P.T. BEE
00200 8710 14TH ST
00210 EXTENSION 18
END OF FILE
?? verify ⑨
?? change/zip code/zip/* ⑩
00110 1544 WILSHIRE ST ZIP 55722   ⑪
00170 POB 55 ZIP 99107
END OF FILE
?? change/extension/ext/* ⑫
00180 EXT 8837
00210 EXT 18      ⑬
END OF FILE
??
```

① User calls XEDIT to edit a file called ADDRESS.

② User issues PRINT command to list the entire file.

③ XEDIT lists file.

④ User issues BRIEF command so that XEDIT will not verify the effects of XEDIT commands.

⑤ User issues CHANGE command; XEDIT does not verify its results.

⑥ User issues another CHANGE command; XEDIT does not verify its results.

⑦ User issues PRINT command.

⑧ XEDIT lists edited file; indicates how the above commands altered the user's file.

⑨ User issues VERIFY command so that XEDIT will verify the effects of XEDIT commands.

⑩ User issues CHANGE command.

⑪ XEDIT verifies which lines were affected by the preceding CHANGE command.

User issues another CHANGE command.

⑬ XEDIT verifies which lines were affected by the preceding CHANGE command.

Figure C-8. VERIFY and BRIEF Commands (Sheet 1 of 2)

```
old,address
READY.
batch
$RFL,0.
/attach,xedit/un=library                ①
/xedit
 XEDIT 3.0
?? print* ②
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T.JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EX6533
00160 O.E. SMITH
00165 APT 23                                  ③
00170 POB 55 ZIP 99107
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST
00210 EXT 18
END OF FILE
?? xchange/zip/zip code/* ④
END OF FILE
?? file,nesf,s ⑤
NEWF     SAVED ⑥
?? xchange/ext/extension/* ⑦
END OF FILE
?? file,,replace ⑧
ADD5     REPLACED ⑨
??
??
```

① User calls XEDIT to edit ADDRESS file.

② User issues PRINT command to list entire file.

③ XEDIT lists file.

④ User issues CHANGE command prefixed by an X; since verify mode is automatically in effect, the X prefix suppresses command verification.

⑤ User issues FILE command to save the editing changes on a new permanent file called NEWF.

⑥ XEDIT indicates that NEWF has been saved in the user's permanent file catalog.

⑦ User issues another CHANGE command with X prefix.

⑧ User issues another FILE command to save these editing changes; since the REPLACE option is selected and the fname parameter defaults to the primary local file, the change from both CHANGE commands now appear on the permanent copy of ADDRESS.

⑨ XEDIT indicates that the edited file has replaced the original file in the user's permanent file catalog.

Figure C-8. FILE Command (Sheet 2 of 2)

```
old,address
READY.
batch
$RFL,0.                                  ①
/attach,xedit/un=library
/xedit
 XEDIT 3.0
?? print* ②
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST
00120 EXT 6533
00160 Q.E. SMITH
^0170 POB 55                                 ③
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST
00210 EXT 18
END OF FILE
?? z$locate/ext/$next-1$+add$xnext2 ④
?? LOCATE/EXT/
00120 EXT 6533
?? NEXT-1                          ⑤
00110 1544 WILSHIRE ST
?? +ADD
? 55722 ⑥
00110 1544 WILSHIRE ST55722
?? XNEXT2 ⑦
?? - ⑧
?? LOCATE/EXT/
00180 EXT 8837
?? NEXT-1
00170 POB 55                   ⑨
?? +ADD
? 55703
00170 POB 5555703
?? XNEXT2
?? - ⑩
?? LOCATE/EXT/
00210 EXT 18
?? NEXT-1
00200 8710 14TH ST                ⑪
?? +ADD
? 55713
00200 8710 14TH ST55713
?? XNEXT2
END OF FILE                                    ⑫
?? y$locate/ext/$next-1$+modify$xnext2
00120 EXT 6533
00110 1544 WILSHIRE ST55722
  00110 1544 WILSHIRE ST55722  ⑬
?                        ↑ zip #⑭
00110 1544 WILSHIRE ST ZIP 55722 ⑮
?? - ⑯
00180 EXT 8837
00170 POB 5555703
  00170 POB 5555703    ⑰
?             ↑ zip #
00170 POB 55 ZIP 55703
?? - ⑱
00210 EXT 18
00200 8710 14TH ST55713
  00200 8710 14TH ST55713      ⑲
?                    ↑ zip #
00200 8710 14TH ST ZIP 55713
END OF FILE
??
```

① User calls XEDIT to edit ADDRESS file.

② User issues PRINT command to list entire file.

③ XEDIT lists file.

④ User issues a Z command which contains 4 component commands; its objective is to add a numeric zip code to the file line containing an address. In this Z command, a $ acts as the delimiter which separates component commands. In addition, the plus (+) prefix on the ADD command tells XEDIT that ADD input data (in this case a zip code) will come from the next entry on the Z command sequence, rather than from normal input mode (a single question mark (?)).

⑤ XEDIT verifies that the first three component commands (LOCATE, NEXT, and ADD) are executed.

⑥ XEDIT verifies how the ADD command affected the file.

⑦ XEDIT verifies the 4th component command (NEXT) is executed.

⑧ User enters a - sign to advance the file pointer (one line by default) and reexecute the preceding Z command; user wants to modify the second address in the file.

⑨ XEDIT reexecutes the Z command sequence.

⑩ User enters a - sign to advance pointer and reexecute Z command; user wants to modify the third address in the file.

⑪ XEDIT reexecutes the Z command sequence.

⑫ User enters a Y command which is made up of 4 component commands; its objective is to add the word ZIP before each numeric zip code in the file. NOTE: Unlike the Z command, the Y command does not list each component command before it is executed.

⑬ XEDIT executes the MODIFY command (which is one of the component Y commands).

⑭ Since the plus (+) prefix was not used on the MODIFY command the user issues his MODIFY directives from normal input mode (a single question mark (?)) to alter line 110.

⑮ XEDIT verifies how these directives affected line 110.

⑯ User enters - sign to reexecute the preceding Y command after advancing the pointer one line; user wants to modify the second address in the file.

⑰ XEDIT reexecutes the Y sequences; user modifies the second address.

⑱ User enters - sign to reexecute Y command and modify the third address in the file.

⑲ XEDIT reexecutes the Y sequence; user modifies the third address.

Figure C-9. Z and Y Commands and + Prefix Character

```
old,address
READY.
batch
$RFL,0.
/attach,xedit/un=library
/xedit
 XEDIT 3.0
?? print* ②
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST
00120 EXT 6533
00160 Q.E. SMITH
00170 POB 55
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST
00210 EXT 18
END OF FILE
?? change/###/***/ ④
*** NAMES/ADDRESSES ARE FICTITIOUS *** ⑤
?? end,,r ⑥
ADDRESS REPLACED ⑦
/lnh     ⑧
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST
00120 EXT 6533
00160 Q.E. SMITH
00170 POB 55
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST
00210 EXT 18
/old,address ⑩
/lnh  ⑪
*** NAMES/ADDRESSES ARE FICTITIOUS **  ⑫
00100 M.T. JONES
00110 1544 WILSHIRE ST
00120 EXT 6533
00160 Q.E.  *INTERRUPTED*  ⑬
stop
 *TERMINATED*
xedit ⑭
 XEDIT 3.0.2
?? print ⑮
*** NAMES/ADDRESSES ARE FICTITIOUS ***
?? change/***/###/ ⑯
### NAMES/ADDRESSES ARE FICTITIOUS ###
?? end,,rl  ⑰
ADDRESS REPLACED ⑱
ADDRESS IS A LOCAL FILE
/lnh ⑲
### NAMES/ADDRESSES ARE FICTITIOUS ### ⑳
00100 M.T. JONES
00110 1544 WsILSHI *TERMINATED* ㉑
old,address ㉒
/lnh  ㉓
### NAMES/ADDRESSES ARE FICTITIOUS ### ㉔
00100 M.T. JONES
00110 1544 WILSHI *INTERRUPTED* ㉕
stop
 *TERMINATED*
```

(Lines old,address through /xedit: ①; lines ### NAMES/ADDRESSES through END OF FILE: ③; lines ### NAMES/ADDRESSES through 00210 EXT 18 after /lnh: ⑨)

① User calls XEDIT to edit ADDRESS file.

② User issues PRINT command to list entire file.

③ XEDIT lists file.

④ User issues CHANGE command to replace each instance of ### with ***.

⑤ XEDIT verifies which line(s) is affected by preceding command.

⑥ User issues END command to terminate XEDIT execution and replace the original ADDRESS file with its edited version.

⑦ XEDIT indicates that the edited file now resides in the user's permanent file catalog.

⑧ User issues LNH command to list contents of the current primary local file.

⑨ NOS lists primary local file; listing indicates that the local file does not contain the XEDIT editing changes. This situation occurs as a result of the user's selection of REPLACE as the mode parameter in the preceding END command.

⑩ User accesses the permanent file version of ADDRESS.

⑪ Userissues LNH command to check its contents.

⑫ NOS listing indicates that the permanent file version of ADDRESS does contain the above XEDIT editing changes.

⑬ User presses break key to terminate listing.

⑭ User calls XEDIT to edit the current primary file (that is, the permanent file version of ADDRESS).

⑮ User issues PRINT command to list the first line in the file; XEDIT prints that line.

⑯ User issues CHANGE command.

⑰ User enters END command to terminate XEDIT execution; specifies RL option as a mode parameter so that the edited file will replace the existing permanent file in the user's catalog and simultaneously also appear as the new primary local file.

⑱ XEDIT indicates that the edited file has replaced the older permanent file and also appears as the primary local file.

⑲ User issues LNH command to list contents of current primary local file.

⑳ NOS lists the local file; listing indicates the editing changes appear on the local file.

㉑ User terminates listing by pressing S key.

㉒ User accesses the current version of ADDRESS permanent file.

㉓ User issues LNH command to check the permanent file's contents.

㉔ NOS lists the permanent file; listing indicates that the edited file replaced the original file.

㉕ User presses break key to terminate listing.

Figure C-10. Terminating XEDIT Execution

This batch job calls XEDIT to create a file called plan according to the directives in the second record of the job. All output goes on the file out which is rewound and printed on the central site line printer.

```
NEWPLAN.
USER(usrname,passwrd,family)
CHARGE(chrgnum,projnum)
ATTACH.XEDIT/UN=LIBRARY
XEDIT(plan,L=out,C)
REWIND(out)
COPYSBF(out)
7/8/9
deftab*
tabs 20 30 40 50 60
input,$
**profit picture
**==============
*1q*2q*3q*4q*total
*--*--*--*--*-----
sales revenue*48,578*51,027*53,598*56,300*$209,503
total expenses*42,273*43,952*45,709*47548* 179,482
operating profit* 6,305* 7,075* 7,889* 8,752*  30,025
$edit
top
print*
end,,rl
6/7/8/9
```

The following is the output from the batch job NEWPLAN.

```
XEDIT 3.0
INPUT
EDIT
                              PROFIT PICTURE
                              ==============
                    1Q        2Q        3Q        4Q        TPTAL
                    --        --        --        --        -----
SALES REVENUE       48,578    51,027    53,598    56,300    $209,503
TOTAL EXPENSES      42,273    43,952    45,709    47,548    $179,482
OPERATING PROFI      6,305     7,075     7,889     8,752    $ 30,025
 END OF FILE
 PLAN     REPLACED
 PLAN     IS A LOCAL FILE
```

Figure C-11. Calling XEDIT Within a Batch Job to Create a File

The batch job FIXPLAN calls XEDIT to edit a permanent file called plan. All XEDIT output goes on a file called out which is rewound and printed on the central site line printer.

```
FIXPLAN.
USER(usrname,passwrd,family)
CHARGE(chrgnum,projnum)
ATTACH,XEDIT/UN=LIBRARY>
ATTACH,XEDIT/UN=LIBRARY.
XEDIT(plan,L=out,P)$CHANGE/30,025/30,021/$TOP$PRINT*$END,,RL
REWIND(out)
COPYSBF(out)
6/7/8/9
```

The following is the listing of the file out.

```
 XEDIT 3.0
OPERATING PROFIT    6,305     7,075     7,889     8,752    $ 30,021
                              PROFIT PICTURE
                              ==============
                    1Q        2Q        3Q        4Q       TOTAL
                    --        --        --        --       -----
SALES REVENUE      48,578    51,027    53,598    56,300    $209,503
TOTAL EXPENSES     42,273    42,952    45,709    47,548    $179,482
OPERATING PROFIT    6,305     7,075     7,889     8,752    $ 30,021
 END OF FILE
 PLAN    REPLACED
 PLAN    IS A LOCAL FILE
```

Figure C-12. Calling XEDIT Within a Batch Job to Edit a File

INDEX

COMMENT SHEET

MANUAL TITLE CDC XEDIT Version 3 Reference Manual

PUBLICATION NO. 60455730 REVISION A

FROM: NAME:

BUSINESS ADDRESS:

CUT ALONG LINE

PRINTED IN U.S.A.

AA3419 REV. 7/75

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY
CONTROL DATA CORPORATION
Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD FOLD