uni-XEDIT

Reference Manual

The Workstation Group, Ltd.
525 Capital Drive
Lake Zurich, IL 60047
Voice: (847) 540-3390
Fax: (847) 540-5629
Email: tech@wrkgrp.com
Web: http://www.wrkgrp.com

Preface

This manual describes the features of the uni-XEDIT® text editor. uni-XEDIT is a Unix implementation of a full screen text editing program, patterned after IBM's VM/CMS System Product Editor.

This manual refers to uni-REXX®. uni-REXX is a Unix implementation of the REXX programming language marketed by The Workstation Group.

Document Number: XEV2-15

Table of Contents

Chapter 1: Introduction 1

Documentation Conventions 2

Chapter 2: Editor Operation 5

Starting uni-XEDIT 5
Edit Ring 7
Unix Wild-Card Characters 9
The Edit Screen 10
Keyboard Functions 13
Program Function Keys 14
Cursor Control Keys 15
Hardware Edit Keys 16
Keyboard Mapping 17
Moving Through the File 19
Prefix Commands 21
Line Commands 22
Targets 23
SET Options 30
Entering Data 31
Saving Data and Ending a Session 34
Multiple Edit Protection 35
File System Full Protection 36
Getting Help 37

Chapter 3: Prefix Commands 39

A (Add) 42
C (Copy) 42
D (Delete) 43
F (Following) 43

I (Insert) 44
M (Move) 44
P (Preceding) 45
S (Show) 45
TABL(Tabline) 46
X (Exclude) 46
“ (Replicate) 47
/ (Position) 48
< (Shift-Left) 48
> (Shift-Right) 49
.xxxx (Set Symbolic Name) 49

Chapter 4: Line Commands 51

ADD 53
ALL 54
BACKWARD 56
BOTTOM 57
CANCEL 58
CAPPEND 59
CDELETE 60
CFIRST 62
CHANGE 63
CINSERT 67
CLAST 68
CLOCATE 69
CMS 72
CMSG 73
COMMAND 74
COMPRESS 75
COPY 77
COVERLAY 79
CREPLACE 80
CURSOR 81
CUT 84
DELETE 85
DOWN 87
DUPLICAT 88
EMSG 90
EXPAND 91
EXTRACT 93
FILE, FFILE 105
FIND 109
FINDUP 110

FORWARD........111
GET........112
HELP........114
HEXTYPE........115
INPUT........116
LEFT........120
LOCATE........121
LOWERCAS........123
MACRO........124
MARK........125
MOVE........126
MSG........127
NEXT........128
PASTE........129
PCOPY........130
PCUT........131
PMOVE........132
POWERINP........133
PRESERVE........134
PUT........135
PUTD........137
QUERY........139
QUIT, QQUIT........140
READ........141
RECOVER........144
REFRESH........145
REPEAT........146
REPLACE........147
RESET........148
RESTORE........149
RGTLEFT........150
RIGHT........151
SAVE, SSAVE........152
SCHANGE........156
SET........157
SHELL........158
SHIFT........160
SORT........162
SOS........164
SPLTJOIN........166
STATUS........167
TOP........168
UP........169
UPPERCAS........170

XEDIT 171
= (Equal Sign) 173
? (Question Mark) 174
! (Exclamation Point) 175
& (Ampersand) 176

Chapter 5: SET Command Options 179

ALT 181
ARBCHAR 182
AUTOSAVE 183
CASE 184
CMDLINE 185
COLOR 186
CTLCHAR 188
CURLINE 191
DISPLAY 192
ENTER 193
FNAME 194
HEX 195
KEYBIND 196
LINEND 203
LRECL 205
MASK 206
MSGLINE 207
MSGMODE 208
NONDISP 209
NULLS 210
NUMBER 211
PENDING 212
PF 213
POINT 215
PREFIX 216
RANGE 218
RECFM 219
RESERVED 220
SELECT 222
SELMODE 223
SCALE 224
SCOPE 225
SCREEN 226
SHADOW 227
SHELLTERM 228
SPAN 229

SPILL 231
STAY 232
SYNONYM....................234
TABLINE....................235
TABS....................236
TRUNC 237
VARBLANK....................238
VERIFY 239
WRAP 240
ZONE 241

Chapter 6: Edit Macros 243

Macro Contents 243
Macro Names 246
Macro Libraries....................247
Macro Execution 248
Customizing the Editor Environment....................251
Command Set Extensions....................254
User Applications Based on uni-XEDIT....................264

Appendix A: Message Summary....................269

Appendix B: System Dependencies....................275

Hewlett Packard 275
IBM Risc System 6000 277
Sun Microsystems 279
System V/386 Workstations....................281

Appendix C: uni-KEY Reference....................283

Starting uni-KEY....................283
Testing Special Keys....................284
Defining Keybinds 285
uni-KEY Limitations....................287

Chapter 1: Introduction

uni-XEDIT from The Workstation Group provides familiar tools and services for individuals migrating from IBM's VM System Product Editor Program (XEDIT) to a distributed processing environment based on Unix workstations. uni-XEDIT Extended also includes an embedded REXX interpreter and provides full capabilities for customization and local extensions to the editor through the implementation of macros or applications based on uni-XEDIT.

The ***uni-XEDIT Reference Manual*** fully describes all features of uni-XEDIT. The companion ***uni-XEDIT Graphical Interface Reference*** describes the features of the graphical user interface available when using uni-XEDIT on an X-windows display.

The ***uni-XEDIT Reference Manual*** is organized as follows:

- **Chapters 1-2** cover documentation conventions and the basic concepts of uni-XEDIT. Individuals unfamiliar with CMS/XEDIT will find these chapters important in understanding the operation of the editor. Chapter 2 also includes information specific to the use of this editor in the Unix environment, which will be valuable to experienced users and novices alike.
- **Chapters 3-5** provide detailed reference for prefix commands, line commands, and SET options, respectively.
- **Chapter 6** describes the use of macros to extend the functionality of uni-XEDIT. It also includes guide-

lines on writing edit macros and some example macros.

- **Appendices A-C** provide information on system dependencies, diagnostic messages, and keyboard mapping.

Documentation Conventions

The following conventions are used throughout this document to facilitate command syntax descriptions.

Uppercase

Uppercase letters indicate commands, keywords, or portions thereof that must be typed exactly as shown. You can type the commands or keywords in any case, regardless of the documentation convention used.

Lowercase

Lowercase letters indicate variable information that you supply. A single character (usually *n*) represents a number that you specify. Other variable data is represented by a descriptive name such as *target* or *string*. Variable data is also italicized to facilitate references to it within descriptive text.

Abbreviations

Some commands and keywords may be abbreviated. When an abbreviation is allowed, this is indicated by uppercase letters, as in the diagram for the DELETE command:

DELete [*target*]

Optional Operands

Many commands have optional keywords or operands. The DELETE example above illustrates this. When an operand may have more than one value, the options are stacked within brackets as in the SET SCREEN command:

```
[SET] SCReen   n      [Horizontal]
                      [Vertical]
```

You actually type only one of the choices – for example SET SCREEN 3 V. Note the use of uppercase letters to indicate the acceptable abbreviation of each optional operand. Initial defaults are shown following the syntax diagram.

Required Operands

Required operands are shown without brackets as in the SHIFT command:

```
SHift     Left     [cols]     [target]
          Right
```

When a required operand may have more than one value, the options are stacked in the same manner as for optional operands. As with optional operands, you type only one of the choices.

Repeating Operands

An ellipsis (. . .) in a syntax diagram indicates that an operand may be repeated one or more times. This is illustrated by the SORT command

```
SORT target   [A]  sort-field-1  [. . . sort-field-n]
              [D]
```

where you may specify multiple sort fields. Do not include the ellipsis when typing your command.

Delimiters

Command operands are normally delimited by a blank space. If any other delimiter is required or optional, it is mentioned specifically in the discussion of an individual command. String targets must be enclosed in delimiters. Examples in this book use the slash (/) to delimit string targets. Any character that is not included in the string may be used as a delimiter for the target.

Chapter 2: Editor Operation

Proficient use of uni-XEDIT requires an understanding of basic editor operation. This chapter provides an overview of uni-XEDIT concepts and features, including a discussion of keyboard functions and keyboard mapping. Subsequent chapters provide detailed information on all uni-XEDIT commands and features. New users of the editor should read this chapter in its entirety. Experienced CMS/XEDIT users may want to pay particular attention to the sections related to Unix-specific issues and keyboard topics.

Starting uni-XEDIT

To start uni-XEDIT from the Unix shell prompt, enter a command in the following form:

```
xe   files   [-Prof profname] [-NOKeybind] [-NOMsg]
             [-NOProf]
   [-help]
   [-l]
```

files is the name of one or more existing files or new files to be edited. The section “Edit Ring” in this chapter discusses simultaneous editing of multiple files.

In the absence of **-Prof** or **-NOProf**, the profile macro “.profile.xedit” is executed. If this exists in the user’s HOME directory, it is used. (The HOME environment variable must be defined.) If no user profile is found, the editor searches for a system profile, which must be named “.profile.xedit”. The system profile must be located in the same directory as the uni-XEDIT binary. If neither a user profile nor a system profile can be located, the edit session starts with no profile.

The **-Prof** operand causes uni-XEDIT to execute the macro ***profname*** instead of ".profile.xedit." ***profname*** is the name of a disk file containing editor line commands or a REXX program. The file name may be any valid Unix filename that conforms to the following rules:

- simple filenames are not valid. If you specify a ***profname*** that does not contain a period (.) in the filename, uni-XEDIT assumes an extension of ".xedit" in searching for the file.
- any filename of the form "***name***.xedit" is valid. You may then specify ***profname*** as "***name***" or as "***name***.xedit".
- complex filenames are all valid. With filenames that contain one or more periods (such as "new.data.xedit"), you **must** specify the entire filename as ***profname***, even if the final component of the name is ".xedit".

The default profile or the specific profile named with the **-Prof** operand applies to all files specified on the "xe" command.

If you specify **-NOProf**, no profile macro is executed. Profile macros are discussed fully in ***Chapter 6: Edit Macros***.

-NOKeybind suppresses default keyboard mappings. Only definitions supplied by the SET KEYBIND command are used. This includes the ENTER key.

-NOMsg suppresses the display of all diagnostic messages as well as messages generated by the CMSG, EMSG, MSG, or QUERY command.

The **-help** option provides help on the syntax for starting uni-XEDIT. The **-help** operand may be abbreviated as **-**. Help for startup syntax also appears if you type the “xe” command without any operands.

The **-l** option provides uni-XEDIT license information. This includes the software serial number, number of licenses, and other information about your license configuration.

Edit Ring

uni-XEDIT allows you to edit multiple files simultaneously within a single edit session. This feature is known as the edit ring.

You may start an edit session with one file or with multiple files. Regardless of how you initiate the session, you may add files to the ring or remove files from the ring at any time.

To start an edit session with multiple files, type

xe ***file1*** ***file2*** ***...*** ***filen*** **[*options*]**

and press Enter. All the files are placed in the edit ring in the order specified, using the default profile or the profile named with the **-Prof** operand. To use different profiles for different files in the ring, you must add each file to the ring separately, as discussed below.

You may use Unix wild-card characters in a filename specification to edit multiple files with similar names. For example,

```
    xe data*
```

places all files beginning with the characters “data” in the edit ring. Files are added to the ring alphabetically. The section entitled “Unix Wild-Card Characters” contains a complete discussion of the use of wild cards in Unix filename specifications. Use of wild card characters in filename specifications is normally valid only when you initially invoke the editor from the Unix shell prompt. Expansion of wild card characters is a function of the shell and not of uni-XEDIT. The uni-XEDIT sample library includes macros that you may use in place of normal file I/O commands if you wish to use shell expansions.

You may add a file to the ring at any time by using the uni-XEDIT line command XEDIT (abbreviated “x”). ***Chapter 4: Line Commands*** includes complete information on the XEDIT line command.)

x ***filename*** **[*options*]**

adds the specified file to the edit ring and displays it as the current file being edited. If no options are specified, the file uses the default profile macro (.profile.xedit).

To view the next file in the ring, use the Xedit line command without operands. The Xedit line command moves in a circular fashion through all files currently in the ring.

To view a specific file in the ring, use

x ***filename***

This locates ***filename*** in the ring and displays it as the current file being edited.

To remove a file from the ring, use the appropriate FILE or QUIT command when that file is currently displayed. The displayed file is removed and the previous file in the ring appears.

To leave the editor without processing each file separately, use the CANCEL command. CANCEL terminates the editing session for all files in the ring, provided there are no unsaved changes. If any file in the ring has unsaved changes, uni-XEDIT displays the file and a message that the file has been changed. The CANCEL operation stops, giving you the opportunity to save the file. You must then re-enter the CANCEL command to proceed.

When files are added to the edit ring without specific profile specifications, the default profile is used. For example, when you type

```
xe  file1  file2  file3
```

the profile macro ".profile.xedit" is used. Include **-Prof** for an alternate profile macro or **-NOProf** for no profile. For example:

```
xe  file1  file2  file3  -prof  different
```

places all three files in the ring using the profile macro "different.xedit".

Unix Wild-Card Characters

For specifying filenames on the "xe" invocation command, you may use the following standard Unix wild card conventions:

*	matches any group of characters
?	matches any single character
[ ]	contains a list of acceptable matches for a single character

These wild card characters may be used singly or in combination to form the appropriate filename pattern. The following table gives examples of how Unix matches wild card characters for file names:

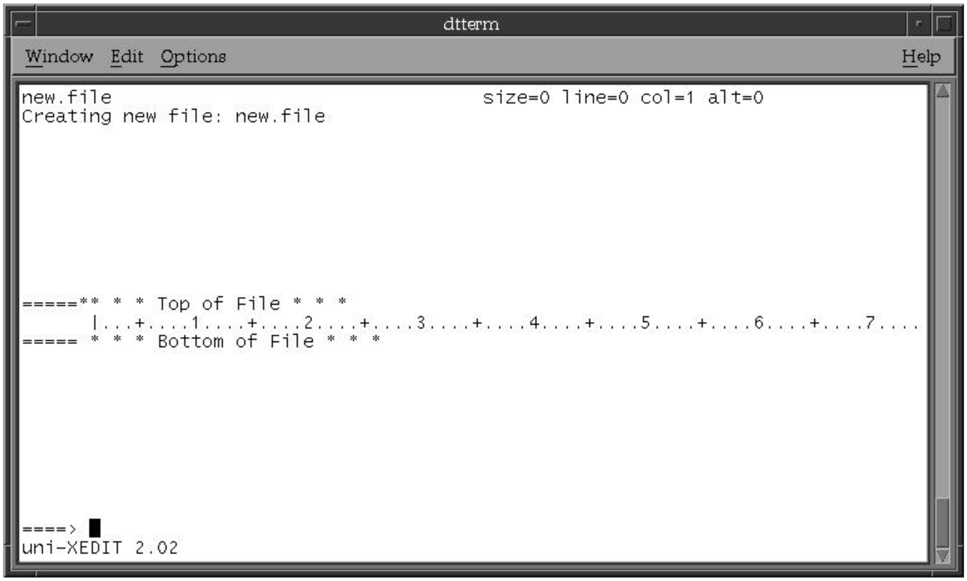
Filename	**Matches**
*	all files in the directory
.	all files in the directory that have an extension; matches myfile.dat and yourfile.text but not myfile or yourfile
sc*	all files in the directory that begin with sc; matches schedule, scan, scan.file, and scan.for.text.
sc*.*	all files in the directory that begin with sc and have an extension; matches scan.file and scan.for.text but not schedule or scan

list?	all files in the directory with 5-character names that begin with “list” and that do not have an extension; matches list1, list2, and list3 but not list1.a, list1.b, or list1.backup
list?.*	matches all files in the directory with 5-character names that begin with “list” and that have an extension; matches list1.a, list1.b, and list1.backup but not list1, list2, or list3
list?.?	all files in the directory with 5-character names that begin with “list” and that have a 1-character extension; matches list1.a and list1.b but not list1, list2, list3, or list1.backup
l[ia]st	all files with 4-character names where the second character may be either “i” or “a”, the other characters as shown, and no extension; matches list and last but not lost or list.today
l[iao]st.*	all files with 4-character names where the second character may be “i”, “a”, or “o”, the other characters as shown, and any extension; matches list.today but not list, last, or lost
l[io]s?*	all files with names where the first character is “l”, the second character may be “i” or “o”, the third character is “s”, the fourth character may be any character, and any number of characters, including an extension, may follow; matches list, listing, listing.file, lisp.program, lost, losalt.diet, and lost.and.found but not last or lesson
l[a-f]s?*	same as the previous pattern except that the second character may be any character from “a” through “f”; matches last and lesson but not list, listing, listing.file, or lost

The Edit Screen

When you start uni-XEDIT, a formatted edit screen appears. The default layout of the edit screen is shown in the figure on the following page. Options of the SET command allow you to alter this layout. Such SET commands are normally included in the uni-XEDIT profile.

The components of the edit screen are described in detail in the paragraphs that follow.



```
new.file                                          size=0 line=0 col=1 alt=0
Creating new file: new.file







=====** * * Top of File * * *
       |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....
===== * * * Bottom of File * * *








====> 
uni-XEDIT 2.02
```

File-id Line

The file-id line is the first line on the screen. It gives the name of the file being edited, the number of lines in the file (size), the line number of the current line, the position of the column pointer, and the number of changes made since the last SAVE (alt). The file-id line is the first line of the screen. You **cannot** change its location.

Message Line

The message line is the second line. It is used for diagnostic messages and for output from the EMSG, MSG and QUERY commands. The default location is the second line of the screen. Use SET MSGLINE to change the location of the message line.

Prefix Area

The prefix area is where you type prefix commands. By default, it is located to the left of each data line and is indicated by five equal signs (=====). Appro-priate combinations of SET PREFIX and SET NUMBER control the location of the prefix area and the characters displayed there (for example, line numbers instead of equal signs).

Current Line

The current line may be a data line, the “Top of File” line, or the “Bottom of File” line. It is indicated by an asterisk between the prefix area and the data. The current line is the reference for operations that occur relative to its location, such as scrolling commands or search operations. By default, it is located near the center of the screen above the scale line. Use SET CURLINE to change the location of the current line.

Scale Line

The scale line is a column position indicator. A vertical bar (|) on the scale line identifies the position of the column pointer, which is the reference for column oriented commands. By default, the scale line appears in the center of the screen. Use SET SCALE to control the presence and the location of the scale line.

Command Line

The command line, indicated by an arrow (====>), is where you type line commands. The HOME key moves the cursor automatically to the beginning of the command line. Its default location is the next to last line of the screen, above the product release indicator. Use SET CMDLINE to change the location of the command line.

Data Area

The remainder of the edit screen is the data area, where data lines of the file being edited appear. In a new file, the data area is empty until you add lines to the file.

Keyboard Functions

uni-XEDIT provides keyboard functions similar to those available on an IBM 3270-type terminal. These include

- Program Function keys (PF keys)
- Cursor control keys
- Hardware edit keys

The keyboards on most terminals used with Unix work-stations do not have a one-to-one correspondence with these 3270-style keys. uni-XEDIT therefore has a generic key sequence defined for each of these keyboard functions. Keyboards with similar keys or with keys that can be mapped to the desired function may use such keys instead of (or in addition to) the generic key sequences.

The generic key sequences may include modifier keys such as Escape (ESC) or Control (CTL). When the key sequence includes ESC, press the Escape key, and then press the key specified. When the key sequence includes CTL, hold down the Control key while you press the second key.

The sections that follow identify the function, the key name (such as PF1), and the generic key sequence. The section on “Keyboard Mapping” discusses how to use similar keys on your keyboard. Workstation- or terminal-specific issues are discussed in ***Appendix B: System Dependencies***. The chapter entitled “A curses Application Primer: Terminal and Keyboard Issues in the Unix World” in the ***TWG System Administrator’s Reference*** includes further details.

Program Function Keys

Program function keys have been defined to perform specific uni-XEDIT commands. The initial uni-XEDIT PF key definitions for PF13-24 are identical to the definitions for PF1-12. Throughout this documentaiton, the key names PF1 through PF12 are used to refer to the default PF key definitions. Although twenty-four PF keys are often available on 3270-style terminals, there may not be sufficient keys available on workstation terminals to define all twenty-four PF keys.

The default PF key definitions and the generic key sequence that activates each one are shown below.

PF Key	Generic Sequence	Definition	Function
PF1	ESC 1	Help	displays help file xe.hlp
PF2	ESC 2		
PF3	ESC 3	Quit	terminates edit session
PF4	ESC 4	Tabkey	moves cursor to next tab position
PF5	ESC 5	Schange 6	selective change, sets PF6
PF6	ESC 6	?	redisplays last command
PF7	ESC 7	Backward	moves current line back in file
PF8	ESC 8	Forward	moves current line forward in file
PF9	ESC 9	=	re-executes last command
PF10	ESC 0	Rgtleft	views data off the screen
PF11	ESC -	Spltjoin	splits or joins lines
PF12	ESC =	Cursor Home	relocates terminal cursor
PF13	ESC !	Help	same as PF1
PF14	ESC @		
PF15	ESC #	Quit	same as PF3
PF16	ESC $	Tabkey	same as PF4
PF17	ESC %	Schange 18	selective change, sets PF18
PF18	ESC ^	?	same as PF6
PF19	ESC &	Backward	same as PF7
PF20	ESC *	Forward	same as PF8
PF21	ESC (	=	same as PF9
PF22	ESC)	Rgtleft	same as PF10
PF23	ESC _	Spltjoin	same as PF11
PF24	ESC +	Cursor Home	same as PF12

You may redefine PF keys to commands that you use more frequently. You may also expand your PF key usage by eliminating the duplication between PF1-12 and PF13-24. Use SET PF*n* to modify your PF key definitions. Operands of SET PF control how command line text is processed relative to the function of the PF key.

On many terminal/workstation combinations, the keys labelled “F1", ”F2", and so on are automatically recognized as PF keys. This depends on the mapping of that terminal’s keyboard that is provided as part of the operating system by the workstation manufacturer. For a full discussion of keyboard mapping, refer to the section entitled “Keyboard Mapping” in this chapter. ***Appendix B: System Dependencies*** contains more detailed information about keyboard mappings that may be needed for specific terminal/workstation combinations. ***Appendix C: uni-KEY Reference*** guides you in using the keyboard test and mapping utility, uni-KEY, which is provided with uni-XEDIT.

Cursor Control Keys

The cursor control keys include the arrow keys, tab, back-tab, and home. In addition, PF12 and PF4 are normally defined as CURSOR HOME and TABKEY, respectively, and may be used to control cursor position.

Arrow keys move the cursor on the screen one character position in the direction indicated. When the cursor reaches the bottom of the screen, it wraps to the top. Similarly, when it reaches the top of the screen, it wraps to the bottom. When the cursor reaches the right edge of the screen, it wraps to the beginning of the next line. Similarly, when the cursor reaches the left edge of the screen, it wraps to the right end of the previous line.

The behavior of the tab key is determined by the setting of the TABS option. With SET TABS OFF (the default), the tab key moves the cursor to the beginning of the next input field. With SET TABS ON, the tab key behaves like the TABKEY key (normally, PF4).

The back-tab key always moves the cursor to the beginning of the previous input field. Its behavior is not affected by SET TABS.

The home key moves the cursor from any area of the edit screen to the upper left input area of the screen.

The key defined as TABKEY (normally PF4) moves the cursor automatically to the next tab stop position. Its behavior is not affected by SET TABS. Use SET TABLINE or the TABL prefix command to display a tab line that shows the tab stops currently in effect.

The key defined as CURSOR HOME (normally PF12) alternates the cursor position between the command line and the last used data or prefix area of the screen.

On most terminal/workstation combinations, the cursor control keys are automatically recognized by uni-XEDIT. In some cases, however, the keyboard mappings provided with the operating system by the workstation manufacturer may be incomplete. If the cursor control keys on your keyboard do not control the cursor correctly, refer to ***Appendix B: System Dependencies*** as well as ***Appendix C: uni-KEY Reference*** for guidance on mapping these keys.

Hardware Edit Keys

Several keys on the 3270-style keyboard assist you in modifying data on the screen. Some have equivalents on workstation terminals while others do not. These keys and their generic key sequences are

Key Name	**Key Sequence**	**Function**
Delete	CTL d	delete one character
Erase-EOF	CTL e	delete to end of input field
Insert	CTL a	enter terminal insert mode
Newline	CTL l	cursor to start of next line
Reset	CTL r	terminate insert mode
Tabchar	CTL t	insert tab character

In many cases, the INSERT and DELETE keys on your terminal keyboard are automatically recognized by uni-XEDIT. The END key may also be recognized as ERASE-EOF.

Few, if any, terminals have keys equivalent to the 3270-style RESET. If there are keys on your keyboard that are not in use for other uni-XEDIT functions, you may choose to map them to one or more of the hardware edit keys. ***Appendix C: uni-KEY Reference*** guides you in using the keyboard test and mapping utility, uni-KEY, to perform such keyboard mappings.

Unlike PC and workstation style keys, the 3270-style INSERT is not a toggle. When you press INSERT, a caret (^) appears near the center of the last line on the screen. This alerts you that you are in insert mode and that characters typed on the keyboard will be inserted into the text at the current cursor position. Press the RESET key to terminate insert mode. The caret indicator disappears when insert mode is terminated. Pressing ENTER or a PF key also terminates insert mode.

On 3270-style terminals, the newline key positions the cursor at the beginning of the next line on the screen. uni-XEDIT supports a keyname “newline”, which may be mapped to any suitable key on your keyboard using SET KEYBIND.

Keyboard Mapping

uni-XEDIT uses the Unix System V curses library for terminal input and output. It determines your terminal type from the TERM environment variable and uses the corresponding terminfo database entry for keyboard mapping. In some cases, the terminfo database provided by the workstation manufacturer may not contain definitions for all the keys used by uni-XEDIT; or the database may contain incorrect definitions for some keys. In other cases, there may be extra keys on the keyboard that you would like to assign to specific editor commands or functions.

Certain workstations and terminals have known terminfo deficiencies. For example, the BACKTAB key does not operate properly on the IBM Risc System 6000 workstation. ***Appendix B: System Dependencies*** includes workstation- and terminal-specific information for platforms on which uni-XEDIT operates. If your particular workstation is not mentioned in this appendix, there are no known system dependencies associated with it.

uni-XEDIT includes a utility, uni-KEY, that allows you to overcome terminfo deficiencies or to use extra keys by mapping a specific key to a uni-XEDIT keyboard function. ***Appendix C: uni-KEY Reference*** provides information about this keybind maintenance utility. uni-KEY can be used in two ways:

- to determine what keycodes are transmitted by individual keys on the keyboard and if the keycode is being interpreted by System V curses from information in the terminfo database for the current terminal type
- to maintain and update the uni-XEDIT profile (.profile.xedit) by generating SET KEYBIND commands to map the desired key to a uni-XEDIT keyboard function or command, thus supplementing or overriding the keybinds provided by curses and terminfo

SET KEYBIND is the recommended method of mapping keyboard definitions for use with uni-XEDIT since it allows custom keyboard definitions for each user on each workstation/terminal combination. Other methods may create incompatibilities in keyboard mappings for other applications in use at your site. ***Chapter 5: SET Command Options*** contains a complete discussion of the use of SET KEYBIND.

Among the alternate methods to map keyboard sequences are changing the master terminfo database for all users or creating a local definition that is accessed through the environment variable TERMINFO. If you choose to do this, the table that follows shows the terminfo keynames that uni-XEDIT uses for each function. You must be sure that the entry for each terminal type to be used by uni-XEDIT has all these key names included and has the actual key sequence that the terminal transmits as the definition for that key name.

Keyboard Function	**Terminfo Keyname**
up arrow	kcuu1
down arrow	kcud1
left arrow	kcub1
right arrow	kcuf1
function keys 1-24	kf1, kf2, kf3, ..., kf24
home	khome
delete	kdch1
insert	kich1
reset	krmir
erase-eof	kend

In some terminfo definitions, kf0 is a synonym for kf10. uni-XEDIT recognizes either keyname.

Moving Through the File

To view data in the file, you may reposition the file display in a variety of ways:

- scrolling one screen at a time
- scrolling by a specified number of lines or columns
- repositioning the current line
- moving immediately to the top or bottom of the file

Scrolling can be performed in either the vertical or horizontal direction. Vertical scrolling always repositions the current line.

You may explicitly reposition the current line with either line commands or the / (slash) prefix command. Line commands are also available to move immediately to the top or bottom of the file.

Scrolling commands are the default definitions for certain PF keys. Other keys, such as Page-Up, may also be automatically mapped to uni-XEDIT scrolling commands. You may use SET PF***n*** or SET KEYBIND to assign frequently-used scrolling commands to PF keys or other key sequences.

The table which follows lists the scrolling line commands, their function, and the default keys associated with each.

Line Command	**Function**	**Default Key**
Backward	back one screen	PF7 Page-Up
Forward	forward one screen	PF8 Page-Down
Rgtleft	right or left one screen (toggles between right and left)	PF10
Top	move to “Top of File”	
Bottom	move to “Bottom of File”	
Up [*n*]	up *n* lines; use * to move to “Top of File”	
Down [*n*]	down *n* lines; use * to move to “Bottom of File’	
Next [*n*]	same as Down	
Right [*n*]	right *n* columns; use 0 to return the display to its original configuration	
Left [*n*]	left *n* columns; use 0 to return the display to its original configuration	
Locate *target*	position the current line according to the target specification; *target* may be an explicit number, a displacement, a symbolic name, or a string	

Prefix Commands

Prefix commands are one- to five-character commands that perform basic editing operations on data lines. You may type a prefix command in any position of the five column prefix area.

A single prefix command operates on the data line where it is typed. If a single prefix command includes an optional numeric operand, it operates on the specified number of lines, beginning with the line where it is typed. Block prefix commands operate on the data lines spanned by the beginning and ending block prefixes.

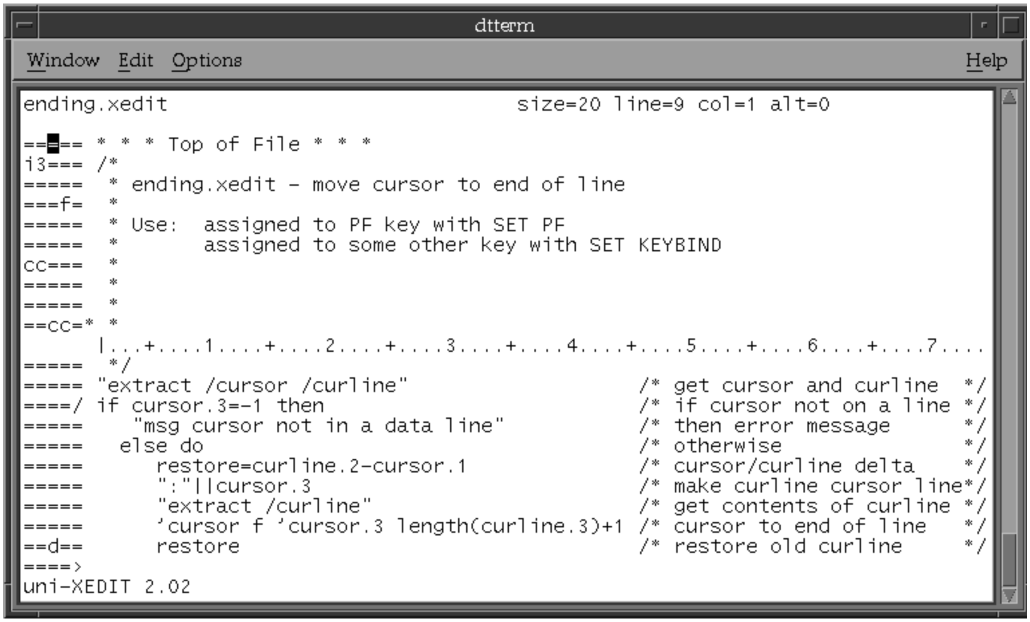
Prefix commands are executed when you press Enter. You may type any number of prefix commands before pressing Enter.

Prefix commands allow you to

- insert blank lines
- delete lines
- copy or replicate lines
- move lines
- exclude lines from display
- re-display lines that have previously been excluded
- position the current line
- shift data to the left or right
- assign symbolic names to data lines
- display a tab line

Prefix commands that copy or move data require the entry of a corresponding prefix to designate the target location for the copy or move. When you enter a copy or move prefix without the required target, the prefix command remains pending until the target prefix is entered. Use the RESET command to remove pending prefixes.

The figure on the following page illustrates a sample uni-XEDIT session in which several prefix commands have been entered simultaneously.



```
ending.xedit                              size=20 line=9 col=1 alt=0

===== * * * Top of File * * *
i3=== /*
=====  * ending.xedit - move cursor to end of line
===f=  *
=====  * Use:  assigned to PF key with SET PF
=====  *       assigned to some other key with SET KEYBIND
cc===  *
=====  *
=====  *
==cc=* *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....
=====  */
===== "extract /cursor /curline"                    /* get cursor and curline  */
====/ if cursor.3=-1 then                           /* if cursor not on a line */
=====    "msg cursor not in a data line"            /* then error message      */
=====   else do                                     /* otherwise               */
=====      restore=curline.2-cursor.1               /* cursor/curline delta    */
=====      ":"||cursor.3                            /* make curline cursor line*/
=====      "extract /curline"                       /* get contents of curline */
=====      'cursor f 'cursor.3 length(curline.3)+1  /* cursor to end of line   */
==d==      restore                                  /* restore old curline     */
====>
uni-XEDIT 2.02
```

Chapter 3: Prefix Commands provides a complete syntax description of all prefix commands and further details on their use.

Line Commands

Line commands perform editing operations, control the screen display, tailor the uni-XEDIT session and perform file management tasks. Enter line commands and their operands on the command line.

If you enter a line command with an operand, you must specify a delimiter between the line command and its first operand and between multiple operands. Command operands are normally delimited by a blank space. If any other delimiter is required or optional, it is mentioned specifically in the discussion of an individual line command. When you enter command operands that contain embedded blanks, the first blank between the command and the operand is the delimiter. Subsequent blanks, however, are part of the operand.

Line command operations occur relative to the current line or the current column pointer. Some line commands reposition the current line or the column pointer.

Two line commands, SET and LOCATE, have special rules that govern their operation. As a result, it is not necessary to explicitly type SET or LOCATE to execute one of these commands. The following rules apply:

- If the text on the command line is a valid target specification, it is assumed to be a LOCATE command. For string targets, the delimiter must be the slash (/).
- If the text on the command line begins with a valid SET command option, it is assumed to be a SET command.

For additional flexibility, you may use SET PF to assign line commands to PF keys.

Chapter 4: Line Commands provides a complete syntax description of all line commands.

Targets

Several uni-XEDIT commands use targets as operands. Targets redefine the current line or the column pointer for specific operations.

A line target *repositions the current line. Line targets are used with numerous commands, including CHANGE, DELETE, LOCATE and SORT.*

*A **column target** repositions the column pointer. Column targets are used only with the CLOCATE and CDELETE commands.*

A target may be

- an explicit number
- a displacement
- a symbolic name
- a string

An ***explicit number target*** *is designated by a colon (:) followed by a number. Examples include:*

`CL :10`	moves the column pointer to column 10
`L :12`	moves the current line to line 12
`:12`	because the LOCATE command itself is optional, identical to L :12

A displacement target moves the column pointer or current line relative to the current position. The displacement is specified as a number or an asterisk (*).

A displacement target may be preceded by a directional indicator to indicate the direction of movement. "+" specifies forward movement and is identical to specifying a number without the directional indicator. "-" speci-fies backward movement. All movement occurs relative to the current line or the column pointer position at the time you enter the command.

The asterisk (*) has special meaning when used as a displacement target. It indicates the maximum possible movement in the direction specified. For column targets, this maximum is determined by the *zone-2* column. (The *zone-2* column is defined in ***Chapter 5: SET Command Options***. By default, it is identical to the truncation column.) Entered alone or with the "+" directional indicator, "*" means the column immediately to the left of the *zone-2* column. With the "-" directional indicator, "*" means the column immediately to the right of the *zone-1* column. For line targets, "*" and "+*" mean the end of the file. "-*" means the beginning of the file.

The following examples illustrate the use of displacement targets with the CLOCATE command:

`CL 6`	moves the column pointer 6 columns right of its current position
`CL +6`	same as CL 6
`CL -4`	moves the column pointer 4 columns left of its current position
`CL *`	moves the column pointer one column left of the *zone-2* column
`CL +*`	same as CL *
`CL -*`	moves the column pointer 1 column right of the *zone-1* column

Examples using line displacement targets include

`L 15`	moves the current line forward 15 lines
`+22`	moves the current line forward 22 lines
`DEL -7`	deletes the current line and the six preceding lines
`DEL *`	deletes from the current line to the end of the file

A **symbolic name** target references a specific line in the file by its symbolic name. Names are assigned to lines using SET POINT or the ***.xxxx*** prefix command. Symbolic name targets are valid only as line targets and have no meaning as column targets.

The following examples illustrate the use of symbolic name targets after you have assigned the name “here” to a line in the file:

`CO 10 .here`	copies 10 lines, beginning with the current line, and places the copies immediately after the line named “here”; the line named “here” becomes the new current line
`.here`	repositions the current line at the line named “here”; identical to L .here
`DEL .here`	deletes all lines from the current line up to, but not including, the line named “here”

A **string target** repositions the column pointer in the column where the first character of the string is found. A string target repositions the current line to the first line where the string is found.

String targets must be enclosed in delimiters. A valid delimiter is any character except +, -, ., or ~. Examples in this book use the slash (/) as the delimiter for string targets.

A string target may be

- a single string
- a “not” string
- multiple strings

A **single string target** is simply a text string enclosed in delimiters. Examples of the use of single string targets include:

`CL /xyz/`	finds the next occurrence of “xyz” after the current column pointer and repositions the column pointer in the location of the “x”. If “xyz” does not occur on the current line, CLOCATE searches through subsequent data lines for the first occurrence of the target.

`DEL /xyz/`	deletes lines beginning with the current line up to, but not including, the first line containing "xyz". The line containing "xyz" becomes the new current line.
`L /xyz/`	repositions the current line on the next line that contains the string "xyz"
`/xyz/`	same as L /xyz/

When you use single string targets, the closing delimiter is not required. `CL /xyz/` is identical to `CL /xyz`.

A string target may be preceded by a directional indicator to indicate the direction of the search. "+" specifies a forward search and is identical to specifying a single string without the directional indicator. "-" specifies a backward search. All searches occur relative to the current line or the column pointer position at the time you enter the command. As examples

`CL +/xyz/`	searches forward from the current column pointer for the string "xyz" and repositions the column pointer
`CL -/xyz/`	searches backward from the current column pointer for the string "xyz" and repositions the column pointer
`-/xyz/`	searches backward from the current line for the string "xyz" and repositions the current line

A **"not" string target** is used to locate the first occurrence of text that does not match the string specified. "Not" is indicated by a tilde (~). Examples of the use of "not" strings include:

`CL ~/xyz/` finds the next occurrence of text that does not match “xyz” and repositions the column pointer in the location of the first non-matching character.

`DEL ~/xyz/`

deletes the current and subsequent lines up to the first line that does not contain “xyz”; repositions the current line on the line without “xyz”

“Not” strings may also have a directional indicator, which must appear before the tilde. Examples of this use are:

`CL +~/xyz/`

searches forward from the current column pointer to locate text that does not match the string “xyz” and repositions the column pointer; identical to CL ~/xyz/

`L -~/xyz/`

searches backward from the current line to locate text that does not match “xyz” and repositions the current line

Multiple string targets are composed of two or more single strings separated by “&”, meaning “and,” or “|”, meaning “or”. With column-oriented commands, multiple string targets may contain only the “|” separator. With line-oriented commands, multiple string targets may contain “&”, “|”, or both.

For multiple string targets that contain only the “|” separator, each line of data is searched for the first string, the next string, and so on. The search does not proceed to a subsequent line of data unless none of the components of the multiple string target exist on the current line. The column pointer or current line is repositioned as soon as any one component of the target is encountered.

For multiple string targets that contain only the “&” separator, each line of data is searched for the first string, the next string, and so on. The search proceeds to subsequent lines unless all of the components of the multiple string target exist on the current line. The current line is repositioned as soon as a line containing all components of the target is encountered.

For multiple string targets that contain both “|” and “&” separators, the logic is evaluated from left to right as it appears on the command. Thus the command

```
L /abc/ & /q23/ | xyz
```

repositions the current line to the first line that contains both “abc” and “q23" or the first line that contains ”xyz", whichever occurs first.

The single string components of a multiple string target may be ordinary single strings or “not” strings. A multiple string target may also be preceded by a directional indicator.

The following examples illustrate the use of multiple string targets:

`CL /xyz/ | /abc/`

searches the current line for “xyz”, beginning at the current column pointer; if not found, searches the current line for “abc”; if not found, repeats the search on subsequent lines until one of the two strings is found; repositions the column pointer at the location of the first character of the string found

`CL /xyz/ | /abc/ | /q23/`

searches forward from the current column pointer for one of the three strings specified; repositions the column pointer when a match is found

`CL -/xyz/ | ~/abc/`

searches backward from the current column pointer for “xyz” or for text that does not match “abc”; repositions the column pointer when a match is found

`DEL /xxx/ & /bbb/`

deletes lines from the current line to the first line that contains both “xxx” and “bbb”; repositions the current line on the line that contains both of these strings

`-~/xxx/ | /zzz/ & ~/bbb/`

searches backward from the current line to locate a line that does not contain “xxx” or a line that contains “zzz” but does not contain “bbb”; repositions the current line on the first line that meets either criterion

Previous examples of string targets have used lowercase strings for illustration. Use SET CASE to control the case sensitivity of string targets.

SET Options

The SET line command supports an array of options through which you control the editing environment. You may

- re-configure the screen display
- define PF keys
- vary the column- and/or line-range for editor operations
- define keyboard mappings
- control case sensitivity for string operands
- assign symbolic names to lines of data
- establish a frequency for automatic saves

and control a variety of other session parameters. Frequently used SET options can be included in your uni-XEDIT profile (.profile.xedit). ***Chapter 5: SET Command Options*** provides a detailed description of each option. This includes the initial default setting and a list of commands affected by the option.

The QUERY line command displays on the Message Line the current setting of SET command options. In edit macros, the EXTRACT command returns this information.

Entering Data

There are several ways to enter new data into a file. You may use

- the Add or Insert Prefix command
- the INPUT line command
- the GET line command
- the ADD line command
- the POWERINP line command

Use the Add or Insert prefix command or the ADD line command to insert blank lines in the file which you may then fill with data. Use the INPUT line command to enter a single line of data into a file or to enter input mode. Use POWERINP for automatic reformatting of data entered in input mode. Use the GET line command to retrieve data from another file.

The Insert or Add prefix command inserts one or more blank lines after the line on which it is typed. Both prefix commands operate identically.

I [*n*]
A [*n*]

n specifies the number of blank lines to be inserted. Insert or Add without an operand inserts a single line. When you press Enter, the editor positions the cursor at column 1 of the first inserted line so you can begin typing new data.

The ADD line command inserts one or more blank lines after the current line.

Add [*n*]

n specifies the number of blank lines to be inserted. ADD without an operand inserts a single line.

The INPUT command has two forms. The first form inserts a single line of data after the current line. The second form enters input mode.

To insert a single line of text immediately following the current line, type

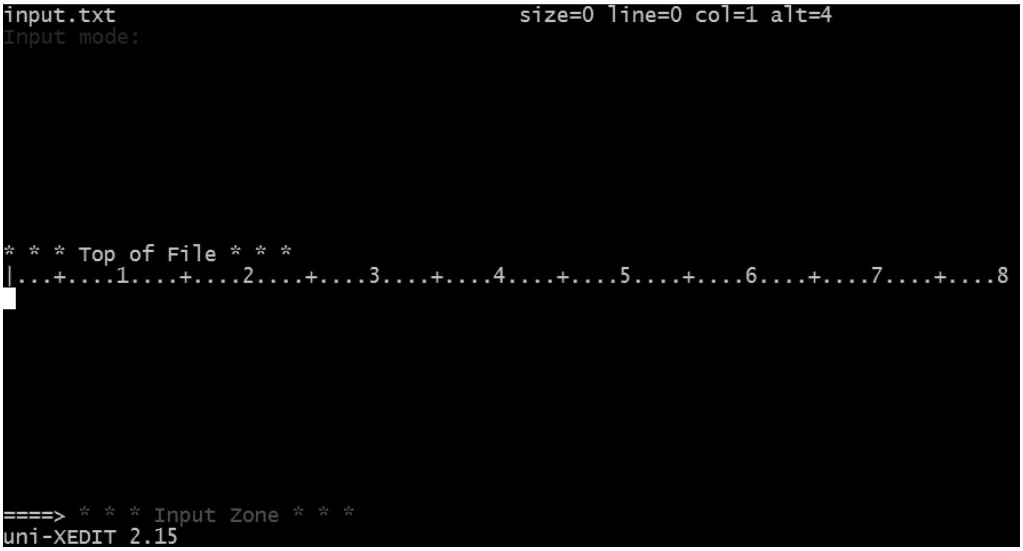
Input *string*

string is the text to be inserted. When you press Enter, the new line contains ***string*** and becomes the new current line.

To enter uni-XEDIT's input mode, use the INPUT command with no operand. Input mode allows you to enter several lines of data immediately following the current line. When you enter input mode, the screen display changes as follows:

- the message "573i Input Mode:" appears on the message line
- *** Input Zone *** appears on the command line
- the prefix area disappears
- all lines following the current line disappear and are replaced by blank lines
- the cursor moves to the first column of the first blank line

The figure on the following page illustrates the uni-XEDIT screen display in input mode.


```
input.txt                               size=0 line=0 col=1 alt=4
Input mode:









* * * Top of File * * *
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8











====> * * * Input Zone * * *
uni-XEDIT 2.15
```

Use the TAB key or the newline key to move from one line to the next. When you are finished entering data, press Enter to “insert” the data and then press Enter again to leave input mode. The current line is automatically repositioned after the last line of data entered.

POWERINP is similar to input mode except that it supports “heads-down” typing and automatically reformats data to insure that line breaks do not occur in the middle of a word.

The GET command retrieves one or more lines from a disk file into the file being edited.

GET *filename* [*n1* [*n2*]]

filename is the name of the disk file from which you wish to retrieve data. ***n1*** is the line number in ***filename*** from which the retrieval begins. ***n2*** is the total number of lines to be retrieved. Data retrieved with GET is placed immediately after the current line at the time the command was entered. When the GET operation is complete, the current line is repositioned on the last line of the retrieved data. The GET command has additional uses which are discussed in more detail in ***Chapter 4: Line Commands***.

Saving Data and Ending a Session

There are several methods of saving data and/or exiting the edit session. You may

- save the data and remain in the edit session
- save the data and exit the edit session
- exit the edit session without saving the data
- request automatic interim saves at a pre-determined frequency (AUTOSAVE)

The SAVE command saves a file to disk and allows you to remain in the edit session.

SAVE [*filename*]

SAVE with no operand saves the current file being edited. The ***filename*** operand allows you to save the file under a different name. SAVE will not over-write ***filename*** if it already exists.

The SSAVE command also saves a file to disk and allows you to remain in the edit session. It differs from SAVE in that it automatically overwrites ***filename*** if it exists.

The FILE command saves a file to disk and exits the edit session.

FILE [*filename*]

FILE with no operand saves the current file being edited. The ***filename*** operand allows you to save the file under a different name. FILE will not overwrite ***filename*** if it already exists.

The FFILE command also saves a file to disk and exits the edit session. It differs from FILE in that it automatically overwrites ***filename*** if it exists.

The QUIT command terminates the edit session as long as you do not have any unsaved changes. To terminate the session even if there are unsaved changes, use QQUIT.

The AUTOSAVE option of the SET command allows you to specify a frequency for automatic interim saves of data.

[SET] AUTOSAVE *n*

n specifies the number of changes allowed between saves. This corresponds to the "alt=" count that appears on the File-id line of the uni-XEDIT screen. The AUTOSAVE feature saves these intermediate versions of the data in a file named "1*nnnnn*.AUTOSAVE" in your HOME directory.

Multiple Edit Protection

The action of the FILE and SAVE commands provides some protection for multiple users editing the same file. In this context, multiple users may be any of the following:

- two or more userids editing the same file with uni-XEDIT
- two or more userids editing the same file with uni-XEDIT and possibly other editors
- one userid editing the same file in two or more windows with uni-XEDIT and possibly other editors

Because of the way this protection is implemented, it is effective across all NFS links as well as on the current workstation.

When you bring a file into uni-XEDIT, the editor captures the current date/time stamp of the file. When you later enter a FILE or SAVE command, the editor checks the date/time stamp of the disk file before performing the save. If the current date/time does not match the original date/time stamp, uni-XEDIT displays the message:

```
1030e File filename has been modified by another
user; use FFILE/SSAVE
```

You then have the option to save the file under another name. This action protects any changes that may have been made by another user. FFILE and SSAVE bypass the date/time stamp checking and overlay the existing file. This action destroys any changes that may have been made by another user.

File protection based on the date/time stamp is only invoked when the filename under which you are saving the file is the same as the filename under which you loaded it. If you specify a different ***filename*** on the FILE or SAVE command or if you use SET FNAME to change the default ***filename*** for a FILE or SAVE command, the usual tests for existence of a file by the new name apply. For these purposes the filenames “junk” and “./junk” are not equivalent even though “junk” resides in the current directory.

File System Full Protection

uni-XEDIT also provides protection against loss of data in the event of file-system-full conditions. Two alternative protection mechanisms are available as well as an option to disable this protection.

During a FILE or SAVE operation, uni-XEDIT protects data by copying the current session to a temporary file, removing the original file, and replacing it with the temporary copy. In the event of a file-system-full condition, a message appears indicating the condition and the location of the preserved copy. In the event that the temporary file cannot be created, you may use the FFILE or SSAVE command to bypass the protection.

The environment variable XETEMP may be used to control the location of the temporary file or to bypass the protection mechanism entirely.

- When XETEMP is not defined, uni-XEDIT writes the temporary file in the same directory as the original file. This is the default behavior.
- When XETEMP is set to a fully qualified directory name, uni-XEDIT writes the temporary file to this directory.
- When XETEMP is set to “NOTEMP”, uni-XEDIT disables file protection completely. No temporary file is created and all attempts to save the file write directly to the original file. (NOTEMP must be specified in uppercase and cannot be abbreviated.)

The most complete protection is provided when XETEMP identifies a directory with ample free space in a file system other than the one where user files typically reside. This directory must have write permission for all users and ample space to save temporary copies of your largest files.

Excellent protection is also provided through the default mechanism, when XETEMP is not set. If a file-system-full condition prevents creation of the temporary file, you may save the file to a different file system by specifying a full path name on the FILE or SAVE command. Alternatively, you may use FFILE or SSAVE to bypass the protection mechanism.

The greatest exposure occurs when XETEMP is set to “NOTEMP”. This disables protection completely and leaves you vulnerable to loss of data. Any attempt to save a file writes directly over the original file. If a file-system-full condition occurs, the output file is truncated at that point and the updated file exists **only** in the current edit session.

Getting Help

Help for uni-XEDIT commands and SET options is available through the HELP line command, which is normally assigned to PF1. When you request help, the uni-XEDIT help file “xe.hlp” is added to the edit ring.

Help is given in a format similar to a quick-reference card. You may use uni-XEDIT commands and PF keys to scroll through the help file or to locate a specific command for which you need help. Press PF3 to exit the help file and return to your current file; or use the XEDIT line command to switch between the help file and other files in the edit ring.

Chapter 3: Prefix Commands

Prefix commands operate on single lines or blocks of lines. Prefix commands may be immediately preceded or followed by a numeric operand indicating the number of lines on which the operation is performed. Prefix commands are typed over any position of the five-column prefix area of the data line(s) to be affected. You may enter the prefix commands in upper or lower case. Prefix commands are executed when you press Enter.

All prefix commands have corresponding line commands that can be used to accomplish the same task. Prefix commands, however, provide a visual approach to file editing and significantly improve the speed and accuracy with which you can modify a file.

The prefix commands are:

ADD	**INSERT**	**“**
COPY	**MOVE**	**/**
DELETE	**PRECEDING**	**<**
EXCLUDE	**SHOW**	**>**
FOLLOWING	**TABL**	***.xxxx***

The prefix area normally appears on the left of the data line and is indicated by five equal signs (=====). Several SET command options are available to change the location or the appearance of the prefix area display. These are:

[SET] PREFIX ON RIGHT
to move the prefix area to the right side of the display

[SET] PREFIX NULLS
to display blanks in the prefix area

[SET] NUMBER ON
to display line numbers in the prefix area

Combinations of these SET options produce a variety of display configurations.

SET PREFIX OFF
removes the prefix area from the display. This gives you six additional columns of data but precludes the use of prefix commands for file modification.

To enter a prefix command, type over the display in the prefix area. Press Enter to execute the command. If you change your mind and do not want to execute a prefix command, you may

- type over the prefix command with a different prefix command
- type over the prefix area with one of the standard display characters (equal signs, numbers, or blanks)
- use the DELETE key to delete prefix command characters
- use ERASE-EOF to clear the contents of the prefix area

You may specify multiple prefix commands before pressing Enter. Prefix commands are executed according to a specific priority. The priority numbers are internal to uni-XEDIT and are used for determining the order of the operations.

Prefix	**Priority**
A,I	60
/	50
“	40
M,C,S,X,<,>	30
D	10

Prefix commands with higher priority numbers are executed before those with lower priorities. Thus, if multiple prefix commands are entered, Insert is performed before Copy or Move, which is performed before Delete.

Some prefix commands require that matching prefixes be entered before the command can be executed. This is true in the case of block commands, where both the beginning and end of the block must be marked, and Copy and Move commands, where a destination must be specified. Such commands remain pending until the matching prefix is entered. Use the RESET line command to remove pending prefixes.

The pages which follow contain a detailed description of each of the prefix commands.

A
(Add)

The Add prefix command inserts one or more blank lines in the file.

A
A*n*
***n*A**

The first form adds a single line immediately following the line on which you type the Add command. The second and third forms add ***n*** lines. After a line or lines are added, the cursor moves to the first column of the first line that was added.

The Add prefix command and the Insert prefix command perform identical functions.

C
(Copy)

The Copy prefix command copies one or more lines to another location in the file. The destination of the copy must be indicated by the F (Following) or P (Preceding) prefix command.

C
C*n*
***n*C**
CC

The first form copies a single line to the specified location. The second and third forms copy ***n*** lines, beginning with the line on which you type the Copy command. A single C, C***n*** or ***n***C command remains pending until a corresponding destination command (F or P) is entered.

The fourth form is used to copy a block of lines. You must type CC on both the first and the last lines of the block to be copied. This is similar to the C***n*** form of the command, but may be more useful when the block is larger than the display screen. In such instances, type CC at one end of the block and scroll forward or backward to locate the other end of the block. A single CC command remains pending until a matching

CC is entered; and a pair of CC commands remains pending until a corresponding destination command is entered.

D
(Delete)

The Delete prefix command deletes one or more lines, or a block of lines.

D
D*n*
***n*D**
DD

The first form deletes a single line. The second and third forms delete ***n*** lines, beginning with the line on which you type the Delete command.

The fourth form is used to delete a block of lines. You must type DD on both the first and the last lines of the block to be deleted. This is similar to the D***n*** form of the command but may be more useful when the block is larger than the display screen. In such instances, type DD at one end of the block and scroll forward or backward to locate the other end of the block. A single DD command remains pending until a matching DD command is entered.

F
(Following)

The Following prefix command specifies a target for a Copy (C) or a Move (M) command.

F

An F command remains pending until you enter a corresponding C or M command.

I (Insert)

The Insert prefix command inserts one or more blank lines in the current file.

I
I*n*
***n*I**

The first form inserts a single line immediately below the line on which you type the Insert command. The second and third forms insert ***n*** lines. After a line or lines are inserted, the cursor moves to the first column of the first line that was inserted.

The Insert prefix command and the Add prefix command perform identical functions.

M (Move)

The Move prefix command moves one or more lines to another location in the current file. The destination of the move must be indicated by the F (Following) or P (Preceding) prefix command.

M
M*n*
***n*M**
MM

The first form moves a single line to the specified location. The second and third forms move ***n*** lines, beginning with the line on which you type the Move command. An M, M***n***, or ***n***M command remains pending until a corresponding destination command (F or P) is entered.

The fourth form is used to move a block of lines. You must type MM on both the first and the last lines of the block to be moved. This is similar to the M***n*** form of the command but may be more useful when the block is larger than the display screen. In such in- stances, type MM at one end of the block and scroll forward or backward to locate the other end of the block. A single MM command remains pending until a matching MM command is entered; and a pair of MM commands remains pending until a corresponding destination command is entered.

P
(Preceding)

The Preceding prefix command specifies a target for a Copy (C) or a Move (M) command.

P

A P command remains pending until you enter a corresponding C or M command.

S
(Show)

The Show prefix command displays one or more lines from a block of excluded lines. The Show prefix command is only valid on the "shadow" line that indicates the number of lines in the excluded block.

S

```
S[+]n
 [-]
nS
```

The first form displays all excluded lines. The second and third forms display ***n*** lines, beginning with the line on which you type the show command.

With the second form, you may optionally specify "+" or "-" to display the first or last ***n*** lines, respectively. The second form also supports the use of asterisk (*) to display all excluded lines. S* is equivalent to S with no operands. "*" is not supported with the optional "+" or "-" operand.

TABL (Tabline)

The TABL prefix command displays a tab line immediately above the line on which TABL is entered.

TABL

The TABL prefix command has the same effect as SET TABL ON ***n*** where ***n*** corresponds to the line number on which you entered the TABL prefix command.

To remove the tab line, delete the line on which it appears or enter the line command SET TABL OFF.

X (Exclude)

The eXclude prefix command temporarily excludes one or more lines from the current display. Lines excluded from display are no longer affected by other edit commands, but they are still in the file.

X
X*n*
***n*X**
XX

The first form excludes a single line. The second and third forms exclude ***n*** lines, beginning with the line on which you type the exclude command.

The fourth form is used to exclude a block of lines. You must type XX on both the first and the last lines of the block to be excluded. This is similar to the X***n*** form of the command but may be more useful when the block is larger than the display screen. In such instances, type XX at one end of the block and scroll forward or backward to locate the other end of the block. A single XX command remains pending until a matching XX command is entered.

Normally, lines excluded from the display are represented by shadow lines of the form

```
------------n Line(s) excluded --------------------
```

where ***n*** is the number of lines excluded from the display. The S (Show) prefix command, which is valid only in the prefix area of shadow lines, allows you to redisplay one or more excluded lines.

SET SHADOW OFF suppresses the display of shadow lines. This also precludes the use of the S prefix command to redisplay excluded data.

“ (Replicate)

The Replicate prefix command replicates a line or block of lines one or more times.

“
“*n*
***n*“**

“”
“”*n*
***n*“”**

The first form replicates a single line immediately below the line on which you typed the Replicate command. The second and third forms replicate the current line ***n*** times, placing the duplicates immediately after the line on which you typed the Replicate command.

The last three forms are used to replicate a block of lines. “” repeats the block once. “”***n*** or ***n***“” repeats it ***n*** times. You must type “” (or “”***n***) on both the first and the last lines of the block to be replicated. You may type “” (or “”***n***) at one end of the block and scroll forward or backward to locate the other end of the block. A single “” command remains pending until a matching “” is entered. It is necessary to provide the ***n*** value on only one of the pair of “” prefix commands.

/
(Position)

The Position prefix command repositions the current line and the current column pointer.

/
/*n*
***n*/**

The first form repositions the current line to the line on which you typed the / command. The column pointer remains in the same position. The second and third forms reposition the current line to the line on which you typed the / command and repositions the column pointer in column ***n***.

<
(Shift-Left)

The Shift Left prefix command shifts data to the left one or more columns on one line or a block of lines. Data shifted to the left past column one (*zone1*) is lost.

<
<*n*
***n*<**

<<
<<*n*
***n*<<**

The first form shifts data left one column on the line on which you typed the < command. The second and third forms shift data left ***n*** columns.

The last three forms are used to shift data left in a block of lines. << shifts data left one column. <<***n*** or ***n***<< shifts data left ***n*** columns. You must type << (or <<***n***) on both the first and the last lines of the block to be shifted. You may type << (or <<***n***) at one end of the block and scroll forward or backward to locate the other end of the block. A single << command remains pending until a matching << is entered. It is necessary to provide the ***n*** value on only one of the pair of << prefix commands.

>
(Shift-Right)

The Shift Right prefix command shifts data to the right one or more columns on one line or a block of lines. Data shifted to the right past the truncation column is lost unless SET SPILL ON or SET SPILL WORD is in effect. In this case, data shifted right past the truncation column is inserted as a new line in the file.

>
>***n***
n>

> **(block shift right_>> (block shift right)>>>**
>>***n***
n>>

The first form shifts data right one column on the line on which you typed the > command. The second and third forms shift data right ***n*** columns.

The last three forms are used to shift data right in a block of lines. >> shifts data right one column. >>***n*** or ***n***>> shifts data right ***n*** columns. You must type >> (or >>***n***) on both the first and the last lines of the block to be shifted. You may type >> (or >>***n***) at one end of the block and scroll forward or backward to locate the other end of the block. A single >> command remains pending until a matching >> is entered. It is necessary to provide the ***n*** value on only one of the pair of >> prefix commands.

.xxxx
(Set Symbolic Name)

The *.xxxx* prefix command assigns a symbolic name to a line. You may then use this symbolic name to reference the line in uni-XEDIT commands that have a target operand.

.*xxxx*

xxxx is a symbolic name for the line. The prefix command must begin with a period and be followed by one to four alphanumeric characters.

The ***.xxxx*** prefix command is identical to the SET POINT command, with the exception that ***.xxxx*** limits the symbolic name to four characters.

The symbolic name is associated with the line through-out the uni-XEDIT session unless you delete it in one of the following ways:

- enter the command SET POINT ***.xxxx*** OFF
- use ***.xxxx*** or SET POINT to assign that name to a different line in the file.

You may assign more than one symbolic name to a single line by typing the ***.xxxx*** prefix more than one time on that line.

Symbolic names are not displayed anywhere on the uni-XEDIT screen and are not saved with the data.

Chapter 4: Line Commands

Line commands operate on the entire file or on a speci-fied range of data in the file. The syntax of a line command is generally in the form of a command followed by required and/or optional operands. Line commands are typed on the command line. You may enter line commands in upper or lower case. Line commands may also be used in a macro. The READ command is used only in edit macros.

The line commands are:

ADD	**CUT**	**MOVE**	**RGTLEFT**
ALL	**DELETE**	**MSG**	**RIGHT**
BACKWARD	**DOWN**	**NEXT**	**SAVE,SSAVE**
BOTTOM	**DUPLICAT**	**PASTE**	**SCHANGE**
CANCEL	**EMSG**	**PCOPY**	**SET**
CAPPEND	**EXPAND**	**PCUT**	**SHELL**
CDELETE	**EXTRACT**	**PMOVE**	**SHIFT**
CFIRST	**FILE,FFILE**	**POWERINP**	**SORT**
CHANGE	**FIND**	**PRESERVE**	**SOS**
CINSERT	**FINDUP**	**PUT**	**SPLTJOIN**
CLAST	**FORWARD**	**PUTD**	**STATUS**
CLOCATE	**GET**	**QUERY**	**TOP**
CMS	**HELP**	**QUIT,QQUIT**	**UP**
CMSG	**HEXTYPE**	**READ**	**UPPERCAS**
COMMAND	**INPUT**	**RECOVER**	**XEDIT**
COMPRESS	**LEFT**	**REFRESH**	**=**
COPY	**LOCATE**	**REPEAT**	**?**
COVERLAY	**LOWERCAS**	**REPLACE**	**!**
CREPLACE	**MACRO**	**RESET**	**&**
CURSOR	**MARK**	**RESTORE**	

Line commands are normally executed by pressing Enter. However, they may also be executed in conjunction with a PF key. SET PF***n*** has an operational operand that allows you to designate whether text on the command line is to be executed when the PF key is pressed.

Many line commands have required or optional target operands. ***Chapter 2: Editor Operation*** contains a complete discussion of targets. The description of each line command that supports a target operand includes specific information relative to its use of targets.

Many line commands perform their operations relative to the current line. The LOCATE command accepts a command as an optional operand, allowing you to position the current line before executing the command.

`LOCATE :10 del 4`

moves the current line to line 10 before deleting 4 lines

`L -6 c/abc/xyz/`

moves the current line six lines toward the top of the file before executing the change command

`/Comments:/ i (For Internal Use)`

moves the current line to the next line containing “Comments:” before inserting a line with the text shown

`.here uppercas *`

moves the current line to the line named “.here” before converting the remainder of the file to upper case

Many line commands are also affected by various SET command options. ***Chapter 5: SET Command Options*** discusses fully all SET command options.

The pages which follow contain a detailed description of each of the line commands.

ADD

The ADD command inserts one or more blank lines immediately after the current line.

ADD [*n*]

With no operands, ADD inserts a single blank line.
n specifies the number of blank lines to be inserted.

Lines added to the file are a permanent part of the file unless explicitly deleted.

Examples:

`add`	inserts one blank line after the current line
`add 10`	inserts 10 blank lines after the current line

ALL

The ALL command is used to limit the display to specific lines, temporarily excluding other lines in the file.

ALL [*target*]

With no operands, ALL displays all lines in the file. This is equivalent to using the S (Show) prefix command

target defines the lines that remain in the display and may be:

- an explicit number
- a displacement
- a symbolic
- name
- a string

Normally, lines excluded from the display are represented by “shadow” lines of the form

```
----------n Line(s) excluded----------
```

where ***n*** is the number of lines excluded from the display. SET SHADOW OFF suppresses the display of shadow lines.

SET command options that may affect the results of the ALL command include

CASE
RANGE
SHADOW
SPAN
TRUNC
VARBLANK
ZONE

Examples:

`ALL /module-b/`
: displays only lines containing the string "module-b"

`ALL /xyz` displays only lines containing the string "xyz"

`ALL /modulea/ | /moduleb/`
: displays lines containing only the string "modulea" or only the string "moduleb" or both strings

`ALL /modulea/ & /moduleb/`
: displays only lines containing both of the strings

`ALL /modulea/ & /moduleb/ | /modulec/`
: displays lines containing both "modulea" and "moduleb" and lines containing "modulec"

`ALL ~/modulea/`
: displays lines that do not contain "modulea"

`ALL /modulea/ | ~/moduleb/`
: displays lines containing "modulea" or not containing "moduleb"

BACKWARD

The BACKWARD command scrolls the screen display toward the beginning of the file. It is normally assigned, without operands, to PF7.

BAckward [*n*]

With no operands, BACKWARD scrolls backward one screen. ***n*** specifies the number of screens to scroll backward. ***n*** may be a number, or it may be * to scroll backward to the beginning of the file.

When you scroll backward to the beginning of the file, the “Top of File” line becomes the current line. If you continue to scroll backward from “Top of File”, the editor wraps to the end of the file making the last line the new current line.

SET command options that may affect the results of the BACKWARD command include

RANGE

Examples:

`BACKWARD`	moves the screen display and the current line backward in the file one screen
`BA 5`	moves the screen display and the current line backward five screens
`BA *`	moves the screen display back to the beginning of the file. The new current line is the “Top of File” line.

BOTTOM

The BOTTOM command scrolls the screen display to the end of the file so that the last line of the file or the last line of the range becomes the current line.

Bottom

When SET RANGE has been used to redefine the bottom line of the file, BOTTOM scrolls to the last line in the defined range.

CANCEL

The CANCEL command terminates the edit session for all files in the edit ring, provided there are no unsaved changes in any of the files.

CANCEL

Entering CANCEL is equivalent to entering QUIT for each file in the ring. If a file in the edit ring has unsaved changes, uni-XEDIT displays the file and the message

```
"577e File has been changed; type QQUIT to  quit
anyway"
```

The CANCEL operation stops, giving you the opporunity to save the file or enter QQUIT. You must re-enter the CANCEL command to proceed.

CAPPEND

The CAPPEND command moves the column pointer to the column beyond the last non-blank character in the current line and appends the specified text.

CAppend [*text*]

With no operands, CAPPEND simply moves the column pointer. When you specify ***text***, it is appended immediately after the last non-blank character in the line.

The first blank delimiter between the CAPPEND command and the ***text*** operand is not part of ***text***. Any subsequent blanks are considered to be part of the operand.

Characters in ***text*** that would extend beyond the truncation column are truncated when SET SPILL OFF is in effect. This is the default. With SET SPILL ON or SET SPILL WORD in effect, characters or words that would extend beyond the truncation column are inserted as one or more new lines in the file.

SET command options that may affect the results of the CAPPEND command include

CASE
SPILL
TRUNC

Examples:

Current line: `Append to`

Command: `cappend end of line.`

(1 space between “cappend” and “end”)

Result: `Append toend of line.`

Command: `cappend  end of line.`

(2 spaces between “cappend” and “end”)

Result: `Append to end of line.`

CDELETE

The CDELETE command deletes text from the current column pointer position up to, but not including, the target.

CDelete ***target***

target defines the column or columns to be deleted. Characters are deleted beginning with the column pointer up to, but not including, the column identified by ***target***. ***target*** may be any of the following:

- an explicit number
- a displacement
- a string

Symbolic name targets are not valid with CDELETE.

SET command options that may affect the results of the CDELETE command include

ARBCHAR
CASE
SPAN
VARBLANK
ZONE

Examples:

`CDELETE :17`	deletes all the characters from the current column pointer up to column 17. The character in column 17 is not deleted.
`CD 7`	deletes seven characters starting with the current column pointer and moving right
`CD -9`	deletes nine characters starting with the current column pointer and moving left

`CD -*` deletes all the characters starting with the current column pointer and moving left to the *zone-1* column

`CD /xyz/` searches forward through the file for the first occurrence of string “xyz” to the right of the column pointer. When “xyz” is found, all characters up to “xyz” are deleted, and the column pointer is positioned at “x”.

`CD /xyz/ | /abc/`
searches for the first occurrence of “xyz” or “abc” to the right of the column pointer and forward through the file. When either “xyz” or “abc” is found, all characters up to the string are deleted, and the column pointer is repositioned to the first character of the string found.

CFIRST

The CFIRST command moves the column pointer left to the *zone-1* column (the leftmost column that you can edit).

CFirst

SET ZONE defines the *zone-1* column. The default is column 1.

SET command options that may affect the results of the CFIRST command include

ZONE

CHANGE

The CHANGE command searches for a string of characters and changes it to a different string.

Change /*string-1*[/*string-2*/ [*target* [*nn* [*nf*]]]]

string-1 is the search string. ***string-2*** is the replacement string. If you omit ***string-2***, a null string is used. A delimiter character must precede, separate, and follow the strings. Valid delimiters are all characters except +, -, ., and ~. Examples in this book use slash (/) as the delimiter.

target defines the lines which are to be changed. Lines are changed beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be any of the following:

- an explicit number
- a displacement
- a symbolic name
- a string

When ***target*** is omitted, CHANGE affects only the current line.

nn specifies the number of occurrences of ***string-1*** to be changed in each line. If you omit ***nn***, only the first occurrence of ***string-1*** in a line is changed. When ***nn*** is specified as an asterisk (*), all occurrences of ***string-1*** in a line are changed.

nf specifies the relative number for the first occurrence of ***string-1*** to be changed in each line. ***nf*** must be a number. If you omit ***nf***, the default is 1. If ***nn*** is *, ***nf*** is invalid.

Selective Change

uni-XEDIT also provides a selective change feature which allows you to view each occurrence of ***string-1*** before confirming the change. This feature uses two

PF keys: one to move to each occurrence of ***string-1*** and the other to perform the change. These are normally PF5 and PF6, respectively.

To use selective change

- Type the CHANGE command on the command line.
- Press PF5 to search for the first (or next) occurrence of ***string-1***.
- Press PF6 to make the change OR
- press PF5 again to search for the next occurrence of string-1.
- Repeat, making the changes you choose, until End of File or the ***target*** on the CHANGE command is located.

SET command options that may affect the results of the CHANGE command include

ARBCHAR
CASE
RANGE
SPILL
STAY
TRUNC
ZONE

Examples:

```
CHANGE /abc/xyz/
```

changes the first occurrence of string “abc” to “xyz” on the current line only

```
C /abc//
```

changes “abc” to a null string on the current line only

`C/abc/def/ 10`

changes the first occurrence of "abc" on each of the next 10 lines, including the current line

`C/abc/def/ +5`

changes the first occurrence of string "abc" on each of the next 5 lines, including the current line

`C/abc/def/ -10`

changes the first occurrence of string "abc" on each of the preceding 10 lines, including the current line

`C/abc/def/ *`

changes the first occurrence of string "abc" on each line from the current line to the end of the file

`C/abc/def/ -*`

changes the first occurrence of string "abc" on each line from the current line to the top of the file

`C/abc/def/ /xyz/`

changes the first occurrence of string "abc" on each line from the current line up to the line containing string "xyz". The line containing "xyz" is not changed.

`C/abc/def/ /xyz/|/hij/`

changes the first occurrence of string "abc" on each line from the current line up to the first line that contains either "xyz" or "hij"

`C/abc/def/ .here`

changes the first occurrence of string "abc" on the line named "here". The symbolic name has been previously defined using SET POINT or the ***.xxxx*** prefix command.

```
C/abc/def/ 10 1
```

changes the first occurrence of “abc” on the current line and the next 10 lines

```
C/abc/def/ 10 *
```

changes all occurrences of “abc” in a line on the current line and the next 10 lines

```
C/abc/def/ * 1
```

changes the first occurrence of “abc” in a line from the current line to the end of the file

```
C/abc/def/ * *
```

changes all occurrences of “abc” in a line from the current line to the end of the file

```
C/abc/def/ /xyz/ 5
```

changes 5 occurrences of “abc” from the current line up to the line containing string “xyz”

```
C/abc/def/ /xyz/ 1 2
```

changes one ocurrence of “abc” beginning with the second occurrence on each line from the current line up to, but not including the line with string “xyz”

```
C/abc/def/ * 2 2
```

changes a maximum of two occurrences of string “abc” beginning with the second occurrence on each line from the current line to the end of the file

CINSERT

The CINSERT command inserts text on the current line beginning at the column pointer. Existing data is automatically shifted to the right.

CInsert *text*

text may include leading and trailing blanks. The first blank after the command is a delimiter and is not part of ***text***. Use the underscore character (_) within ***text*** to include blanks within or at the end of the text to be inserted.

Characters in ***text*** which would extend beyond the truncation column are truncated when SET SPILL OFF is in effect. This is the default. With SET SPILL ON or SET SPILL WORD in effect, characters or words that would extend beyond the truncation column are inserted as one or more new lines in the file.

SET command options that may affect the results of the CINSERT command include

CASE
SPILL
TRUNC

Examples:

Current line:	`This belongs here.`
Column pointer:	above the blank between `This` and `belongs.`
Command:	`cinsert insertext`
Results:	`This insertextbelongs here.`
Command:	`cinsert insertext_` ***(blank after insertext)***
Results:	`This insertext belongs here.`

CLAST

The CLAST command moves the column pointer to the *zone-2* column (the rightmost column that you can edit).

CLAst

The default *zone-2* column is the truncation column, which is 4096 by default.

SET command options which may affect the results of the CLAST command include

TRUNC
ZONE

CLOCATE

The CLOCATE command moves the column pointer to a specified column target.

CLocate *target*

target defines the new column pointer position and may be

- an explicit number
- a displacement
- a string

Symbolic name targets are not valid with CLOCATE.

SET command options which may affect the results of the CLOCATE command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
WRAP
ZONE

Examples:

`CLOCATE :14`	moves the column pointer to column 14
`CL 7`	moves the column pointer seven columns to the right
`CL +8`	moves the column pointer eight columns to the right
`CL -9`	moves the column pointer nine columns to the left

`CL *`	moves the column pointer to the column immediately before the *zone-2* column
`CL +*`	same as the previous example
`CL -*`	moves the pointer to the position immediately after the *zone-1* column
`CL /xyz/`	search for the first occurrence of string “xyz” to the right of the column pointer and forward through the file. When “xyz” is found, the column pointer is positioned at “x”. Repeating this command searches in the same direction for the next occurrence and again moves the column pointer.
`CL -/abc/`	searches for the first occurrence of “abc” to the left of the column pointer and backward through the file. When “abc” is found, the column pointer is positioned at “a”. Repeating this command searches in the same direction for the next occurrence and again moves the column pointer.
`CL /mno/ | /xyz/`	searches the current line to the right of the column pointer first for “mno” and then for “xyz”. If neither string is found, the search continues in the same order on the next line forward in the file. When one of the strings is found, the column pointer moves to the first character of the string.
`CL -/abc/ | /def/`	similar to the previous example except that the search proceeds backward through the file if neither string is found on the current line

```
CL -/abc/ | /def/ | /ghi/
```

searches backward looking first for “abc”, then “def”, then “ghi”

```
CL ~/bananas/
```

search for any string that is not “bananas” and place the column pointer at the beginning of that string

```
CL -~/bananas/
```

same as the previous example except that the search proceeds backward through the file

```
CL +~/bananas/ | ~/oranges/
```

searches forward, one line at a time, first for any string that is not “bananas” or not “oranges”. The search is for the absence of “bananas” or the absence of “oranges”, in that order. If a line contains “peaches”, the search stops there since one of the target conditions has been met.

```
CL +~/bananas/ | /apples/
```

searches forward for any string that is not “bananas” or that is “apples”. The search is for the absence of “bananas” or the presence of apples, in that order. If a line contains “peaches”, the search stops there since one of the target conditions has been met.

CMS

The CMS command provides access to the Unix shell for command execution. The specific shell used is determined by the current setting of the SHELL environment variable.

CMS [*string*]

With no operands, CMS exits to the Unix shell for execution of one or more shell commands. In graphical mode, a new terminal window appears for shell command execution. To return to the uni-XEDIT session, you may either type “exit” at the Unix prompt or close the terminal window. In character mode, shell command I/O occurs at the bottom of the current window, which scrolls upward as commands are executed. To return to the uni-XEDIT session, type “exit” at the Unix prompt. The message

```
Press ENTER to return to the editor
```

appears. Press Enter to resume the edit session.

The ***string*** operand is a valid Unix command that is passed to the Bourne shell for execution. The message

```
Press ENTER to return to the editor
```

appears after the output of the Unix command.

The CMS command is synonymous with the SHELL and ! commands.

SET command options that may affect the results of the SHELL command include

SHELLTERM (graphical mode only)

Example:

`CMS ls`	produces a file listing for the current directory, followed by the message “Press ENTER to return to the editor”

CMSG

The CMSG command displays a message on the command line (instead of the message line). It does not generate an alarm. CMSG allows you to display messages even when MSGLINE is set to OFF.

CMSG [*text*]

With no operands, the command line is cleared and the cursor positioned at the beginning of the command line. ***text*** specifies the characters to be displayed on the command line. ***text*** may include leading, trailing, or embedded blanks. All blanks are preserved.

SET command options that may affect the results of the CMSG command include

MSGMODE

Example:

```
CMSG Error condition detected
```

displays the message on the command line

COMMAND

COMMAND executes the specified uni-XEDIT command without first searching for a synonym or macro of the same name.

COMMAND ***string***

string may be any valid uni-XEDIT command and may include the command's operands.

SET command options associated with the COMMAND command include

SYNONYM

Example:

```
COMMAND bot
```

executes the uni-XEDIT BOTTOM command rather than the macro bot.xedit or the synonym "bot" defined by SET SYNONYM

COMPRESS

The COMPRESS command prepares one or more lines in a file for automatic repositioning of data according to new tab stop settings. After a line is compressed, you may set new tab stops with SET TABS and reposition the data with EXPAND.

COMPRESS [*target*]

With no operands, COMPRESS operates on the current line. ***target*** defines the number of lines to be compressed. Lines are compressed beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options associated with the COMPRESS command include

TABS

Examples:

`COMPRESS` compresses the current line only

`COMPRESS 5`
compresses 5 lines beginning with the current line

`COMPRESS -10`
compresses the current line and the preceding 9 lines

`COMPRESS *`
compresses all remaining lines in the file beginning with the current line

`COMPRESS .here`

compresses the current line and all lines up to, but not including, the line named "here". The symbolic name has been previously assigned using SET POINT or the ***.xxxx*** prefix command.

`COMPRESS /abc/`

compresses the current line and all lines up to, but not including, the first line containing "abc"

COPY

The COPY command copies one or more lines, beginning with the current line to a specified location.

COpy [*target-1* *target-2*]

With no operands, COPY copies the current text selection to the clipboard.

target-1 defines the lines to be copied. All lines from the current line up to, but not including, the line identified by ***target-1*** are copied.

target-2 defines the location for the copy. Copied lines are placed immediately after the line identified by ***target-2***.

Both ***target-1*** and ***target-2*** may be any of the following:

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the COPY command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

```
COPY 10 :25
```

copies 10 lines, beginning with the current line, and places the copy immediately after line 25

`CO 2 -1`	copies 2 lines, beginning with the current line, placing the copy after the line immediately preceding the current line
`CO :9 :25`	copies lines beginning with the current line up to, but not including, line 9 and places the copy immediately after line 25
`CO 2 *`	copies 2 lines, beginning with the current line, to the bottom of the file
`CO .here .there`	copies lines beginning with the current line up to, but not including, the line named "here" and places the copy immediately after the line named "there." The symbolic names "here" and "there" have been previously assigned using SET POINT or the ***.xxxx*** prefix command.
`CO /xyz/ :1`	copies lines beginning with the current line up to, but not including, the first line that contains "xyz" and places the copy immediately after line 1
`CO -/data/ /comments/`	copies lines beginning with the current line backward to, but not including, the first preceding line that contains "data"; places the copy immediately after the first line following the current line that contains "comments"

COVERLAY

The COVERLAY command allows you to selectively replace one or more characters in the current line, beginning at the column pointer.

COVerlay ***text***

text is a group of characters that replaces characters in the corresponding position in the current line. Blank characters in ***text*** indicate that the corresponding characters in the current line are not to be overlaid. Use the underscore character (_) to indicate that a blank is to overlay an existing character. You therefore cannot use COVERLAY to insert the underscore character into an existing line.

When SET SPILL OFF is in effect, characters that would extend beyond the truncation column are truncated. With SET SPILL ON or SET SPILL WORD in effect, characters or words extending beyond the truncation column are inserted into the file as one or more new lines.

SET command options that may affect the results of the COVERLAY command include

CASE
SPILL
TRUNC

Examples:

Current line:	`hohoho`
Command:	`COVERLAY my_ my` (1 space after "_")
Results:	`my omy`
Command:	`COV my_  my` (2 spaces after "_")
Results:	`my ohmy`

CREPLACE

The CREPLACE command replaces text on the current line beginning at the column pointer.

CReplace [*text*]

With no operands, CREPLACE replaces the current line with a blank line. ***text*** specifies the characters to overlay existing text. The first blank between the CREPLACE command and ***text*** is a delimiter and is not part of ***text***. Other blanks preceding or within ***text*** replace existing characters in the data.

When SET SPILL OFF is in effect, characters that extend beyond the truncation column are truncated. With SET SPILL ON or SET SPILL WORD, characters or words that would extend beyond the truncation column are inserted as one or more new lines in the file.

SET command options that may affect the results of the CREPLACE command include

CASE
SPILL
TRUNC

Examples:

Current line:	`This old text belongs here.`
Column pointer:	above the “o” in “old”
Command:	`creplace new data`
Results:	`This new data belongs here.`

CURSOR

The CURSOR command moves the cursor to a specified position on the edit screen. In interactive usage, CURSOR is used from a PF key. CURSOR HOME is normally assigned to PF12.

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

The **CMdline** operand moves the cursor to a specified ***colno*** of the command line. When ***colno*** is omitted, CURSOR CMdline moves the cursor to column 1 of the command line.

The **Column** operand moves the cursor to the current line and current column pointer position.

The **File** operand moves the cursor to the specified ***lineno*** relative to the beginning of the file. ***lineno*** must be a line that currently appears on the screen display. ***colno*** specifies the column number where the cursor is placed. When ***colno*** is omitted, CURSOR File moves the cursor to column 1 of ***lineno***.

The **Home** operand alternates the cursor position between the command line and the data area of the screen. When the cursor is on the command line, CURSOR Home moves it to its previous position in the data area of the screen. When the cursor is in the data area, CURSOR Home moves it to the command line.

The **Screen** operand moves the cursor to the specified ***lineno*** relative to the beginning of the screen. The File-id line, message line, scale line, “Top of File” and “Bottom of File” lines are included in counting screen line numbers. ***colno*** specifies the screen column where the cursor is placed. The prefix area of a line is included in counting screen column numbers. When

colno is omitted, CURSOR Screen moves the cursor to column 1 of ***lineno***.

The **Priority** operand assigns a priority to the CURSOR command. This allows cursor positioning to be determined on the basis of pending input at the time the CURSOR command is executed.

n must be a number from 0 to 256. Larger numbers have higher priority.

When **Priority** is omitted, the CURSOR command determines the position of the cursor regardless of any priority associated with pending input. When the **Priority** operand is used with CURSOR, the cursor is positioned according to the highest priority assignment associated with pending input or the current CURSOR command.

Examples - Interactive Use:

PF12 assignment:	`CURSOR HOME`
Pending input:	I prefix command (The cursor normally moves to the first input position of the newly inserted line when this prefix command is executed.)
Action:	Press PF12
Cursor position:	beginning of command line

PF12 assignment:	CUR H P 20
Pending input:	I prefix command (The priority associated with the I prefix is 60.)
Action:	Press PF12
Cursor position:	first input position of newly inserted line

PF12 assignment:	`<F102P10.5M>CUR H P 100`
Pending input:	I prefix command
Action:	Press PF12
Cursor position:	beginning of command line

Examples - Macro Use:

`CURSOR FILE 7`	moves the cursor to column 1 of line 7 of the file
`CUR F 10 20`	moves the cursor to column 20 on line 10 of the file
`CUR S 10 10`	moves the cursor to the tenth line on the screen, starting from the file-id line; the cursor is positioned 10 characters from the beginning of this line, starting with the prefix area
`CUR C`	moves the cursor to the current line without changing its column position

CUT

The CUT command copies the current text selection to the clipboard and deletes it from the file being edited. For interactive use in character mode, CUT may be assigned to a PF key. It may also be used in a macro.

CUT

DELETE

The DELETE command removes one or more lines from the file.

DELete [*target*]

With no operands, DELETE removes only the current line. ***target*** defines the number of lines to delete. Lines are deleted beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the DELETE command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

`DELETE`	deletes the current line only
`DEL 7`	deletes the current line and the next six lines
`DEL :18`	deletes the current line and all other lines up to, but not including, line 18
`DEL -*`	deletes the current line and all previous lines in the file

`DEL /xyz/`

deletes the current line and all lines up to, but not including, the first line that contains “xyz”

`DEL /abc/|/xyz/`

deletes the current line and all lines up to the first line containing either “abc” or “xyz”"

DOWN

The DOWN command advances the current line toward the end of the file.

Down [*n*]

With no operands, DOWN advances the current line one line. ***n*** specifies the number of lines to advance. ***n*** may be a number, or it may be * to advance to the end of the file.

The DOWN command is synonymous with the NEXT command.

SET command options that may affect the results of the DOWN command include

RANGE

Examples:

`DOWN`	moves the current line down one line
`D 5`	moves the current line down five lines
`D *`	moves the current line to the “Bottom of File” or “Bottom of Range” line

DUPLICAT

The DUPLICAT command replicates one line or a group of lines one or more times. Duplication begins with the current line. Duplicate lines are placed immediately after the original line(s). The last line replicated becomes the new current line.

DUPlicat [*n*] [*target*]

With no operands, DUPLICAT replicates the current line once.

n specifies the number of times a line or lines are to be duplicated. ***n*** must be a number.

target defines the number of lines to be replicated. Lines are duplicated beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the DUPLICAT command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

`DUPLICAT`	duplicates the current line once
`DUP 3 12`	makes three duplicates of the current line and the eleven lines after it
`DUP 6`	duplicates the current line six times
`DUP 2 /xyz/`	duplicates twice the current line and all lines up to, but not including, the first line that contains “xyz”

EMSG

The EMSG command displays a message at the terminal and generates an alarm. The message appears on the message line of the screen. EMSG is normally used in edit macros.

EMSG [*text*]

With no operands, EMSG generates an alarm and displays blanks on the message line. ***text*** specifies the characters to be displayed on the message line. ***text*** may include leading, trailing, or embedded blanks. Leading and trailing blanks are discarded.

When MSGMODE is OFF, no message appears and no alarm is generated.

If EMSG appears in a macro immediately after a command that results in a message display, uni-XEDIT must display both messages. You must press Enter to refresh the screen display when this condition occurs.

SET command options that may affect the results of the EMSG command include

MSGLINE
MSGMODE

Example:

```
EMSG Error condition detected
```

displays the message on the message line and generates an alarm

EXPAND

The EXPAND command repositions compressed data containing tab characters on one or more lines in a file according to the current tab stop settings.

EXPAND [*target*]

With no operands, EXPAND operates only on the current line. ***target*** defines the number of lines to be expanded. Lines are expanded beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the EXPAND command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK

Examples:

`EXPAND`	expands the current line only
`EXPAND 5`	expands 5 lines beginning with the current line
`EXPAND -5`	expands the current line and the preceding 4 lines
`EXPAND *`	expands all remaining lines in the file beginning with the current line

```
EXPAND /abc/
```

expands the current line and all lines up to, but not including, the first line containing “abc”

```
EXPAND ~/abc/
```

expands the current line and all lines up to, but not including, the first line that does not contain “abc”

EXTRACT

The EXTRACT command returns the current setting of the SET command options and other information about the file being edited. It is valid only in an edit macro.

EXTract /*option* [/*option* . . .]

option may be any of the following keywords:

ALT	**MASK**	**SIZE**
ARBCHAR	**MSGLINE**	**SPAN**
AUTOSAVE	**MSGMODE**	**SPILL**
CASE	**NBFILE**	**STAY**
CMDLINE	**NONDISP**	**SYNCSCROLL**
COLUMN	**NULLS**	**SYNONYM**
CTLCHAR	**NUMBER**	**TABLINE**
CURLINE	**PENDING**	**TABS**
CURSOR	**PF**	**TOF**
DISPLAY	**PREFIX**	**TOL**
ENTER	**RANGE**	**TRUNC**
EOF	**RECFM**	**UIMODE**
EOL	**RESERVED**	**VARBLANK**
FNAME	**RING**	**VERIFY**
FLSCREEN	**SCALE**	**VERSION**
LINE	**SELECT**	**WRAP**
LINEND	**SELMODE**	**ZONE**
LRECL	**SHADOW**	
LSCREEN	**SHELLTERM**	

You may specify one or more ***options*** on an EXTRACT command. A delimiter character **must** precede each ***option***. A valid delimiter is any character except +, **-**, ., or ~. The examples in this manual use slash (/) as the standard delimiter.

EXTRACT returns its information as a variable in the macro. For each ***option*** specified, at least two variables are returned. Variables names are of the form

option.n

where ***n*** is a number from 0 to the number of variables returned by ***option***.

option.0 is always set to the number of additional variables returned by this option. ***option***.1 through ***option.n*** contain the specific settings for ***option*** in the current file.

The list that follows describes the variables returned for each ***option*** of EXTRACT. Options are shown using standard syntax conventions (minimum abbreviations in upper case) followed by the values of the returned variables. ***Chapter 5: SET Command Options*** provides details on option settings and their effects on the file being edited.

ALT

ALT.0	2
ALT.1	number of alterations since last AUTOSAVE
ALT.2	number of alterations since last SAVE

ARBchar

ARBCHAR.0	2
ARBCHAR.1	ON or OFF
ARBCHAR.2	*arbitrary character*

AUtosave

AUTOSAVE.0	3
AUTOSAVE.1	*frequency* or OFF
AUTOSAVE.2	file AUTOSAVEd
AUTOSAVE.3	changes since last AUTOSAVE

CASE

CASE.0	2
CASE.1	UPPER or MIXED
CASE.2	RESPECT or IGNORE

CMDline

CMDLINE.0	2
CMDLINE.1	ON, OFF, TOP, or BOTTOM
CMDLINE.2	screen line number of the command line

COLumn

COLUMN.0	1
COLUMN.1	current column number of the column pointer

CTLchar

CTLCHAR.0	3
CTLCHAR.1	ON or OFF
CTLCHAR.2	the character defined as the escape character
CTLCHAR.3	list of all defined control characters

CURline

CURLINE.0	5
CURLINE.1	the text entered for the positioning operand of SET CURLINE
CURLINE.2	screen line number of the current line
CURLINE.3	line contents
CURLINE.4	ON (curline changed or inserted in this session) OFF (curline not changed or inserted in session)
CURLINE.5	OLD (curline not inserted in this session) NEW (curline inserted in this session) OLD CHANGED or NEW CHANGED (curline changed during this session)

CURsor

CURSOR.0	9
CURSOR.1	line number of cursor on screen
CURSOR.2	column number of cursor on screen
CURSOR.3	line number of cursor in file
CURSOR.4	column number of cursor in file
CURSOR.5	original screen line position (initiall CURSOR.1)
CURSOR.6	original screen column position (initiall CURSOR.2)
CURSOR.7	original file line position (initial CURSOR.3)
CURSOR.8	original file column position (initial CURSOR.4)
CURSOR.9	the highest cursor priority

DISPlay

DISPLAY.0	2
DISPLAY.1	lowest selection level in the display
DISPLAY.2	highest selection level in the display

ENTer

ENTER.0	2
ENTER.1	BEFORE, AFTER, ONLY, or IGNORE
ENTER.2	*string* (ENTER key definition)

EOF

EOF.0	1
EOF.1	ON (current line is "Bottom of File") OFF (current line is not "Bottom of File")

EOL

EOL.0	1
EOL.1	ON (column pointer at end of line (*zone-2*+1)) OFF (column pointer not at end of line)

FLscreen

FLSCREEN.0	2
FLSCREEN.1	line number in file of first line displayed
FLSCREEN.2	line number in file of last line displayed

FName

FNAME.0	1
FNAME.1	*filename*

LIne

LINE.0	1
LINE.1	line number in the file of the current line

LINENd

LINEND.0	2
LINEND.1	ON or OFF
LINEND.2	logical line-end character

LRecl

LRECL.0	1
LRECL.1	*n* (logical record length)

LSCreen

Use EXTRACT /LSCREEN to determine the dimensions of the total screen (virtual screen) or the screen split in which the macro runs (logical screen). With SET SCREEN 1 (the default display), the logical screen dimensions are identical to the virtual screen dimensions. With SET SCREEN *n* (n>1), there are *n* logical screens within the virtual screen.

For purposes of determining the number of lines in the virtual screen, the product banner line at the bottom of the screen is not included. For purposes of determining the number of lines in a logical screen, the file-id line that appears at the top of each logical screen is counted only for the topmost logical screen.

Under X-Windows, Open Look, Motif, or other windowing environments, the virtual screen is the window in which the current uni-XEDIT session is being run. For VT100's or other non-windowed environments, the virtual screen is the entire screen display.

LSCREEN.0	6
LSCREEN.1	logical screen: number of lines
LSCREEN.2	logical screen: number of columns
LSCREEN.3	logical screen: line number of upper left corner
LSCREEN.4	logical screen: column number of upper left corner
LSCREEN.5	virtual screen: number of lines
LSCREEN.6	virtual screen: number of columns

MASK

MASK.0	1
MASK.1	current mask setting

MSGLine

MSGLINE.0	2
MSGLINE.1	ON or OFF
MSGLINE.2	the text entered for the positioning operand of SET MSGLINE

MSGMode

MSGMODE.0	2
MSGMODE.1	ON or OFF
MSGMODE.2	LONG or SHORT

NBFile

NBFILE.0	1
NBFILE.1	number of files in the edit ring

NONDisp

NONDISP.0	1
NONDISP.1	current “non-display” character

NULls

NULLS.0	1
NULLS.1	ON or OFF

NUMber

NUMBER.0	1
NUMBER.1	ON or OFF

PENDing

PENDING.0	7
PENDING.1	line number in the file
PENDING.2	*newname* (name entered in prefix area); defined by SET PREFIX SYNONYM
PENDING.3	*oldname* (original name, after synonym resolution)
PENDING.4	BLOCK (if prefix block entry is located in the pending list)
	NULL (if a prefix block entry is not located in the pending list)
PENDING.5	*op1* (the first operand of the prefix, if it exists)
	NULL (if no prefix operand exists)
PENDING.6	*op2* (the second operand of the prefix, if it exists)
	NULL (if no prefix operand exists)
PENDING.7	*op3* (the third operand of the prefix, if it exists)
	NULL (if no prefix operand exists)

PF*n*

PF*n*.0	2
PF*n*.1	BEFORE, AFTER, ONLY, or IGNORE
PF*n*.2	*string* (PF*n* key definition)

PF or PF*

Definitions in same format as above for all PF keys in order (1-24). If a key is undefined, PF*n*.0 = 0 with no following variables. Typically variables returned are as shown below:

PF1.0	2
PF1.1	BEFORE
PF1.2	HELP
PF2.0	2
PF2.1	BEFORE
PF2.2	(*PF2 not defined*)
PF3.0	2
PF3.1	BEFORE
PF3.2	QUIT
:	
:	

PREfix

PREFIX.0	2
PREFIX.1	ON, OFF, or NULL
PREFIX.2	LEFT or RIGHT

RANge

RANGE.0	2
RANGE.1	*n* (number of first line in range)
RANGE.2	*n* (number of last line in range)

RECFm

RECFM.0	1
RECFM.1	V, F, FP, or VP Internal status only, all UNIX files are variable (V).

RESERved

RESERVED.0	1
RESERVED.1	list of reserved line numbers, separated by spaces

RESERved *

Returns a value for each reserved line. Although color, extended highlighting, and use of alternate program symbol sets is not supported in this release of uni-XEDIT, the syntax is supported for macro compatibility.

RESERVED.0	*n* (number of variables returned)
RESERVED.1	*linenum color exthi ps# high|nohigh text*
RESERVED.2	*linenum color exthi ps# high/nohigh text*
:	
:	
RESERVED.*n*	*linenum color exthi ps# high|nohigh text*

RING

RING.0	*n* (number of variables returned)
RING.1	number of files currently in the ring
RING.2	descriptor line for first file
RING.3	descriptor line for second file
:	
:	
RING.*n*	descriptor line for last file

Descriptor lines are of the form

filename DATA A1 V *lr* Trunc=*q* Size=*r* Line=*s* Col=*t* Alt=*u*

where

filename	name of the file
DATA A1 V	filetype, filemode, and record format strings provided for compatibility with macros ported from the mainframe
lr	current value of the LRECL setting
q	current TRUNC setting
r	number of lines in the file
s	current line
t	current column position
u	number of changes to file since last SAVE or AUTOSAVE

SCALe

SCALE.0	3
SCALE.1	ON or OFF
SCALE.2	the text entered as the positioning operand of SET SCALE
SCALE.3	line number on the screen

SELECT

SELECT.0	2
SELECT.1	selection level assigned to the current line
SELECT.2	maximum selection level within the file

SELMODE

SELMODE.0	1
SELMODE.1	current text selection mode

SHADow

SHADOW.0	1
SHADOW.1	ON or OFF

SHELLTERM

SHELLTERM.0	1
SHELLTERM.1	current setting of SHELLTERM

SIZe

SIZE.0	1
SIZE.1	number of lines in the file

SPAN

SPAN.0	3
SPAN.1	ON or OFF
SPAN.2	BLANK or NOBLANK
SPAN.3	*n* (number of target lines)

SPILL

SPILL.0	1
SPILL.1	OFF, ON, or WORD

STAY

STAY.0	1
STAY.1	ON or OFF

SYNCSCROLL

SYNCSCROLL.0	2
SYNCSCROLL.1	horizontal synchronized scroll setting
SYNCSCROLL.2	vertical synchronized scroll setting

SYNonym

SYNONYM.0	1
SYNONYM.1	ON or OFF

SYNonym *name*

SYNONYM.0	4
SYNONYM.1	*newname*
SYNONYM.2	*n* (minimum abbreviation length)
SYNONYM.3	*oldname*
SYNONYM.4	NULL

SYNonym *

(All definitions of all synonyms)

SYNONYM.0	*n* (number of synonyms)
SYNONYM.1	*newname abbrev. oldname*
SYNONYM.2	*newname abbrev. oldname*
:	
:	

TABLine

TABLINE.0	3
TABLINE.1	ON or OFF
TABLINE.2	the text entered for the positioning operand of SET TABLINE
TABLINE.3	screen line number or the TABLINE

TABS

TABS.0	2
TABS.1	ON or OFF
TABS.2	tab stop settings, separated by spaces

TOF

TOF.0	1
TOF.1	ON (current line is "Top of File") OFF (current line is not "Top of File")

TOL

TOL.0	1
TOL.1	ON (when cursor is at start of current line (*zone-1*-1)) OFF (when cursor is not at start of current line)

TRunc

TRUNC.0	1
TRUNC.1	*n* (truncation column number)

UIMODE

UIMODE.0	1
UIMODE.1	user interface mode ("CHARACTER" or "GRAPHICAL")

VARblank

VARBLANK.0	1
VARBLANK.1	OFF or ON

Verify

VERIFY.0	2
VERIFY.1	OFF or ON
VERIFY.2	*startcol endcol*

VERSION

VERSION.0	1
VERSION.1	current version of uni-XEDIT

WRap

WRAP.0	1
WRAP.1	OFF or ON

Zone

ZONE.0	2
ZONE.1	*zone-1* (left)
ZONE.2	*zone-2* (right)

Examples:

`EXTRACT /PF13`

returns the definition of PF13 if it has been defined, otherwise it returns PF13.0 = 0

`EXTRACT /case /shadow`

returns the variables for both the CASE and SHADOW options

FILE, FFILE

The FILE command writes the current file to disk and exits the edit session.

FILE [*filename*]

When ***filename*** is omitted, FILE writes the file to disk using the filename displayed on the file-id line. Use the ***filename*** operand to saved the file under a different name.

When you enter a FILE command, uni-XEDIT checks for two conditions that would inhibit saving the file:

- another user has changed the file since your edit session started (for saving under the current name)
- a file by the name ***filename*** already exists (for saving under a different name)

In either case, uni-XEDIT displays an appropriate message, does not save the file, and remains in the edit session. In the first instance, the message is

```
1030e File filename has been modified by another
user; use FFILE/SSAVE
```

In the second instance, the message is

```
594e File filename already exists, use FFILE/SSAVE
```

To save the file and exit the edit session without regard to the current status of the disk file, use the FFILE command.

FFile [*filename*]

SET command options that may affect the results of the FILE or FFILE command include

FNAME
LRECL

Multiple Edit Protection

The action of the FILE command provides some protection for multiple users editing the same file. In this context, multiple users may be any of the following:

- two or more userids editing the same file with uni-XEDIT
- two or more userids editing the same file with uni-XEDIT and possibly other editors
- one userid editing the same file in two or more windows with uni-XEDIT and possibly other editors

Because of the way this protection is implemented, it is effective across all NFS links as well as on the current workstation.

When you bring a file into uni-XEDIT, the editor captures the “last modified” date/time stamp of the file. When you later enter a FILE command, the editor again checks the “last modified” date/time stamp of the disk file before performing the save. If the current date/time stamp does not match the original date/time, uni-XEDIT displays message number 1030e.

You then have the option to save the file under another name. This action protects any changes that may have been made by another user. FFILE bypasses the date/time stamp checking and overwrites the existing file. This action destroys any changes that may have been made by another user.

File protection based on the date/time stamp is only invoked when the name under which you are saving the file is the same as the name under which you loaded it. If you specify a different filename on the FILE command or if you use SET FNAME to change the default name for a FILE command, the usual tests for existence of a file by the new name apply. For these purposes, the filenames “junk” and “./junk” are not equivalent even though “junk” resides in the current directory.

File-System-Full Protection

uni-XEDIT also provides protection against loss of data in the event of file-system-full conditions. Two alternative protection mechanisms are available as well as an option to disable this protection.

During a FILE or SAVE operation, uni-XEDIT protects data by copying the current edit session to a temporary file, removing the original file, and replacing it with the temporary copy. In the event of a file-system-full condition, a message appears indicating the condition and the location of the preserved copy. In the event that the temporary file cannot be created, you may use the FFILE or SSAVE command to bypass this protection mechanism.

The environment variable XETEMP may be used to control the location of the temporary file or to bypass the protection mechanism entirely.

- When XETEMP is not defined, uni-XEDIT writes the temporary file in the same directory as the original file. This is the default behavior.
- When XETEMP is set to a fully qualified directory name, uni-XEDIT writes the temporary file to this directory.
- When XETEMP is set to “NOTEMP”, uni-XEDIT disables file protection completely. No temporary file is created and all attempts to save the file write directly to the original file. (NOTEMP must be specified in uppercase and cannot be abbreviated.)

The most complete protection is provided when XETEMP identifies a directory with ample free space in a file system other than the one where user files typically reside. This directory must have write permission for all users and ample space to save temporary copies of your largest files.

Excellent protection is also provided through the default mechanism, when XETEMP is not set. If a file system-full condition prevents creation of the temporary file, you may save the file to a different file system by specifying a full path name on the FILE or SAVE command. Alternatively, you may use FFILE or SSAVE to bypass the protection mechanism.

The greatest exposure occurs when XETEMP is set to “NOTEMP”. This disables protection completely and leaves you vulnerable to loss of data. Any attempt to save a file writes directly over the original file. If a file-system-full condition occurs, the output file is truncated at that point and the updated file exists **only** in the current edit session.

Examples:

`FILE`
: writes the edited file to disk using the name displayed on the File-id line

`FFILE`
: writes the edited file to disk using the name displayed on the File-id line; overwrites the current file even if it has been modified by another user since your edit session began

`FILE acctbal`
: writes the current file to disk using the name “acctbal” but does not overlay “acctbal” if it already exists

`FFILE acctbal`
: writes the current file to disk as “acctbal” even if a file by this name already exists

FIND

The FIND command searches forward in the file for the first line that begins with the specified text. The search starts with the line after the current line.

Find ***text***

text may be any character string. Embedded blanks in ***text*** are treated as arbitrary characters and match anything that may be in the file in the positions where the blanks appear.

When SET WRAP OFF is in effect, FIND stops searching for ***text*** when the end of the file is reached. When SET WRAP ON is in effect, the search wraps to the top of the file and continues until it returns to the original starting point.

SET command options that may affect the results of the FIND command include

- CASE
- RANGE
- STAY
- WRAP

Examples:

`FIND data`	searches forward for a line that begins with “data”
`F 1   a`	searches forward for a line with “1" in column 1, anything in columns 2, 3, and 4, and ”a" in column 5
`F   job`	searches forward for a line with anything in the first two columns and “job” in columns 3 through 5

FINDUP

The FINDUP command searches backward in the file for the first line that begins with the specified text. The search starts with the line before the current line.

FINDUp ***text***

An alternate abbreviation for FINDUP is

FUP

text may be any character string. Embedded blanks in ***text*** are treated as arbitrary characters and match anything that may be in the file in the positions where the blanks appear.

When SET WRAP OFF is in effect, FINDUP stops searching for ***text*** when the top of the file is reached. When SET WRAP ON is in effect, the search wraps to the bottom of the file and continues until it returns to the original starting point.

SET command options that may affect the results of the FINDUP command include

- CASE
- RANGE
- STAY
- WRAP

Examples:

`FINDUP data`	searches backward for a line that begins with “data”
`FINDU 1   a`	searches backward for a line with “1" in column 1, anything in columns 2-4, and ”a" in column 5
`FUP   job`	searches backward for a line with anything in columns 1-2 and “job” in 3-5

FORWARD

The FORWARD command scrolls the screen display toward the end of the file. It is normally assigned, without operands, to PF8.

FOrward [*n*]

With no operands, FORWARD scrolls forward one screen. ***n*** specifies the number of screens to scroll forward. ***n*** may be a number, or it may be * to scroll forward to the end of the file.

When you scroll forward to the end of the file, the “Bottom of File” line becomes the current line. If you continue to scroll forward from “Bottom of File”, the editor wraps to the top of the file, making the first line the new current line.

SET command options that may affect the results of the FORWARD command include

RANGE

Examples:

`FORWARD`	moves the screen display and the current line forward in the file one screen
`FO 5`	moves the screen display and the current line forward five screens
`FO *`	moves the screen display forward to the end of the file. The “Bottom of File” or “Bottom of Range” line is the new current line.

GET

The GET command retrieves lines from a disk file or from the temporary data buffer into the file you are currently editing. Retrieved lines are placed immediately after the current line.

GET [*filename* [*firstline* [*numlines*]]]

With no operands, GET retrieves data from the temporary data buffer. Such data must have been previously placed in the buffer with a PUT or PUTD command.

filename is the name of a disk file from which to retrieve one or more lines of data. When you specify ***filename*** alone, the entire file is inserted after the current line in the file you are editing.

firstline is the line number of the first line in the disk file to be retrieved. When you specify only ***filename*** and ***firstline***, GET retrieves all lines from ***firstline*** to the end of ***filename***.

numlines specifies the number of lines to retrieve. This limits the amount of data retrieved from ***filename***.

When SET SPILL OFF is in effect, any retrieved line that extends beyond the truncation column of the file you are currently editing is truncated. When SET SPILL ON or SET SPILL WORD is in effect, any text on any retrieved line that would extend beyond the truncation column is inserted as one or more new lines in the file.

SET command options that may affect the results of the GET command include

SPILL
TRUNC

Examples:

`GET fileabc`

retrieves all lines from “fileabc”, placing them after the current line in the file being edited

`GET fileabc 17`

retrieves line 17 and all subsequent lines from “fileabc”, placing them after the current line in the file being edited

`GET fileabc 17 5`

retrieves lines 17 through 21 of “fileabc”

`GET` retrieves all lines from the temporary data buffer

HELP

The HELP command displays information about uni-XEDIT commands and options. The HELP command is normally assigned to PF1.

Help

When you request help, the uni-XEDIT help file, "xe.hlp", is added to the current edit ring. This file presents information about commands and options in a format similar to a quick-reference card. This file **must** reside in a directory that is included in your PATH environment variable. The component of your PATH that identifies the location of the uni-XEDIT help file must begin with a slash (/).

Because you are viewing the help file within the editor, you can navigate it as you would any other file. Use the PF keys to scroll forward and backward through the file or other uni-XEDIT commands to locate specific text or to position the display.

To switch between the help file and other files in the edit ring, use the XEDIT line command.

To exit from the help file, press PF3.

HEXTYPE

The HEXTYPE command displays one or more lines in both hexadecimal and ASCII.

HEXtype [*target*]

With no operand, HEXTYPE displays only the current line. ***target*** defines the number of lines to display beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

The results of the HEXTYPE command are displayed outside the normal uni-XEDIT screen configuration. At the end of the display, the message

```
Press ENTER to return to the editor
```

appears.

SET command options that may affect the results of the HEXTYPE command include

ARBCHAR
CASE
RANGE
VARBLANK

Examples:

`HEXTYPE :5`	displays from the current line up to, but not including, line 5
`HEX 2`	displays the current line and the following line
`HEX *`	displays all lines from the current line to the end of the file

INPUT

The INPUT command is used to insert a single line of text into the file or to enter uni-XEDIT's input mode.

Input [*string*]

With no operands, the INPUT command enters input mode. This is discussed in detail later in this section.

string is the text to be inserted into the file. When ***string*** is specified, a new line containing ***string*** is inserted immediately after the current line. This new line then becomes the current line. The first blank space between the INPUT command and ***string*** is the delimiter and is not part of ***string***. Subsequent blanks are considered to be part of ***string***.

When SET SPILL OFF is in effect, characters in ***string*** that would extend beyond the truncation column are truncated. When SET SPILL ON or SET SPILL WORD is in effect, characters beyond the truncation column are inserted as one or more additional lines in the file.

SET command options that may affect the results of the INPUT command include

CASE
SPILL
TRUNC

Input Mode

When you omit ***string***, you enter uni-XEDIT's input mode. The figure on the following page shows the screen changes that occur in input mode:

- the message “573i Input mode:” appears on the message line
- the text * * * Input Zone * * * appears on the command line
- the prefix area disappears
- all lines following the current line disappear and are replaced by blank lines
- the cursor moves to the first column of the first blank line, ready for input

When you have entered all the data you want on the first line, use the TAB or newline key to move to the next blank line.

When you have filled all lines in the input zone, press Enter. The lines you have entered move to the top half of the screen. The last line you typed becomes the current line. The cursor moves to column one of the next blank line in the input zone, ready for you to continue entering data.

To exit input mode, press Enter twice. The lines you typed move to the top half of the screen. The last line you typed becomes the current line. uni-XEDIT restores the screen to normal edit mode with the cursor on the command line.

You may vary the size of the input zone by using SET CURLINE. Moving the current line nearer the top of the screen results in a larger input zone. Moving it toward the bottom of the screen creates a smaller input zone.

When you use PF keys in input mode, all the lines you typed move to the top half of the screen; the last line typed becomes the current line and uni-XEDIT automatically returns to edit mode. The command associated with the PF key is then executed and uni-XEDIT automatically returns to input mode, placing the cursor at the first input field in the input zone.

You should also be aware of the effect that the command assigned to the PF key may have on input that follows use of that PF key. For example, if you press the PF key assigned to FORWARD, uni-XEDIT scrolls forward one screen in the file and leaves you in input mode. However, the input area is now one screen nearer the end of the file. Data you enter after pressing the PF key is inserted into the file at the new location and **not** immediately after data you entered before pressing the PF key.

The SPLTJOIN command (normally assigned to PF11) is useful in input mode but functions differently than it does in edit mode. You cannot use SPLTJOIN to join together two lines in the input zone. You can, however, use SPLTJOIN to split a line in the input zone. The data that is split moves to a hold queue and is retained there until you press SPLTJOIN to join it to a different line in the input zone or until you press Enter. If you press SPLTJOIN, the data from the queue is appended at the current cursor position. If you press Enter, the data from the queue is inserted as a new line in the file immediately after the last line you typed.

The SPLTJOIN hold queue can contain more than one line of data at a time. Each time you use SPLTJOIN to split a line, the new data is added as a new line at the top of the queue. The lines in the queue are retrieved in a last-in-first-out fashion. When you press SPLTJOIN, the most recent data added to the queue is joined at the current cursor position. When you press Enter, all lines in the queue are added to the end of your input.

LEFT

The LEFT command shifts the edit screen display one or more columns to the left, allowing you to view data that is to the left of the current display.

LEft [*n*]

With no operands, LEFT shifts the display by one column. ***n*** specifies the number of columns to shift left. When ***n*** is specified as zero, LEFT positions the display so that column 1 of the data is in the leftmost display position.

Examples:

`LEFT 10`	moves the display left 10 columns
`L`	moves the display left 1 column
`L 0`	displays the left-most screen of data, beginning with column 1

LOCATE

The LOCATE command moves the current line to the line defined by a target. Optionally, you may specify a command to be executed after the target is located.

[Locate] ***target*** **[*command*]**

The LOCATE command itself is optional. When you specify only ***target*** on the command line, LOCATE is assumed.

target defines the new current line position and may be

- an explicit number
- a displacement
- a symbolic name
- a string

command may be any valid line command. When ***command*** is specified, LOCATE executes the command after ***target*** is located.

SET command options that may affect the results of the LOCATE command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
WRAP
ZONE

Examples:

`LOCATE :7`	makes line 7 the current line
`:17`	makes line 17 the current line
`L 15`	moves the current line forward 15 lines

`-19`	moves the current line backward 19 lines
`L *`	moves the current line to “Bottom of File” or “Bottom of Range”
`-*`	moves the current line to “Top of File” or “Top of Range”
`L /abc/`	searches forward for the first occurrence of “abc” and positions the current line there
`/abc/`	same as the previous example
`-/xyz/`	searches backward for the first occurrence of “xyz” and positions the current line there
`/plums/ | /pears/`	moves the current line forward to the first line that contains either “plums” or “pears”
`/plums/ & /pears/`	moves the current line forward to the first line that contains both “plums” and “pears”
`L -/plums/ & /pears/ | /prunes/`	searches backward for a line that contains “plums” and “pears” or a line that contains “prunes”; positions the current line there
`-~/plums/`	searches backward for a line that does not contain “plums” and positions the current line there
`/abc/ DEL`	moves the current line to the first line that contains “abc” and then executes the DELETE command, deleting that line

LOWERCAS

The LOWERCAS command converts uppercase letters to lowercase letters on one or more lines in the file.

LOWercas [*target*]

With no operands, LOWERCAS converts only the current line.

target defines the range of lines to convert. LOWERCAS converts lines beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the LOWERCAS command include

ARBCHAR
CASE
RANGE
STAY
TRUNC
ZONE

Examples:

`LOWERCAS`	converts the current line only
`LOW 5`	converts the current and next four lines
`LOW *`	converts the current line and all subsequent lines to the end of the file
`LOW -/xy/`	converts the current and all previous lines up to, but not including, the line containing “xy”

MACRO

The MACRO command causes uni-XEDIT to execute the specified macro. The MACRO command may be issued from the command line or from within another macro.

MACRO ***name***

The ***name*** operand specifies the name of the macro file followed by its arguments, if any. The first word of ***name*** is always interpreted as the name of the macro file. Subsequent words of ***name*** are interpreted as arguments for the macro.

You may execute macros from the command line or within another macro by specifying only the macro name. If the macro name is identical to a uni-XEDIT line command or a synonym that you have defined, you must use the MACRO command to explicitly invoke the macro.

When processing macros, uni-XEDIT searches first the current working directory and then in any directories specified in the Unix environment variable XEDITPATH. If the macro is not found, the following message appears

```
"542e No such subcommand: name"
```

When the macro to be executed does not reside in the current working directory or in a directory specified in XEDITPATH, you must use MACRO followed by the fully qualified path of the macro file.

Chapter 6: Edit Macros contains additional details about creating and using uni-XEDIT macros, including information on setting the XEDITPATH environment variable.

MARK

The MARK command selects or de-selects text. For interactive use in character mode, it is normally assigned to a PF key. It may also be used within a macro.

```
MARK     [ADJUST]
         [CANCEL]
```

With no operands, MARK specifies the beginning and end points of selected text. Position the cursor at the beginning of the text to be selected and issue a MARK command. Then reposition the cursor to the end of the text to be selected and issue another MARK command.

ADJUST changes the size of an existing selection.

CANCEL de-selects the current text selection.

SET command options that may affect the results of the MARK command include

SELMODE

MOVE

The MOVE command moves one or more lines, beginning with the current line, to a specified location. The last line moved becomes the new current line.

MOve ***target-1*** ***target-2***

target-1 defines the number of lines to be moved. All lines from the current line up to, but not including, the line identified by ***target-1*** are moved.

target-2 defines the location for the move. Moved lines are placed immediately after the line identified by ***target-2***.

Both ***target-1*** and ***target-2*** may be any of the following

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the MOVE command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

```
MOVE 10 :25
```

moves ten lines, beginning with the current line, to the location following line 25

```
MO 2 /xyz/
```

moves two lines to the location following the first line that contains "xyz"

MSG

The MSG command displays a message on the uni-XEDIT message line. Unlike EMSG, it does not generate an alarm.

MSG [*text*]

With no operands, MSG displays blanks on the message line. ***text*** specifies the characters to be displayed on the message line. ***text*** may include leading, trailing, or embedded blanks. Leading and trailing blanks are discarded.

When MSGMODE is OFF, no message appears.

If MSG appears in a macro immediately after a command that results in a message display, uni-XEDIT must display both messages. You must press Enter to refresh the screen display when this condition occurs.

SET command options that may affect the results of the MSG command include

MSGLINE
MSGMODE

Example:

```
MSG Error condition detected
```

displays the message on the message line

NEXT

NEXT advances the current line toward the end of the file.

Next [*n*]

With no operands, NEXT advances the current line one line. ***n*** specifies the number of lines to advance. ***n*** may be a number, or it may be * to advance to the end of the file.

The NEXT command is synonymous with the DOWN command.

SET command options that may affect the results of the NEXT command include

RANGE

Examples:

`NEXT`	moves the current line down one line
`N 5`	moves the current line down five lines
`N *`	moves the current line to the “Bottom of File” or “Bottom of Range” line

PASTE

The PASTE command copies text from the clipboard into the file at the current cursor position. For interactive use in character mode, it is normally assigned to a PF key. It may also be used in a macro.

PASTE

Use the COPY or CUT command to place text on the clipboard.

PCOPY

The PCOPY command copies the current text selection to the current cursor position. For interactive use in character mode, it is normally assigned to a PF key. It may also be used in a macro.

PCOPY

PCUT

The PCUT command deletes the current text selection. For interactive use in character mode, it is normally assigned to a PF key. It may also be used in a macro.

PCUT

PMOVE

The PMOVE command moves the current text selection to the current cursor position. For interactive use in character mode, it is normally assigned to a PF key. It may also be used in a macro.

```
PMOVE
```

POWERINP

The POWERINP command is used to enter uni-XEDIT's input mode. It is similar to the INPUT command except that data is automatically reformatted when you exit input mode.

POWERINP

Changes to the screen display and the behavior of function keys and the Enter key are identical to those for input mode using the INPUT command. POWERINP, however, provides a "heads-down" input facility that automatically wraps characters when typing reaches the right edge of the screen. When you exit input mode, text is automatically reformatted such that line breaks occur at the spaces between words.

PRESERVE

The PRESERVE command saves the settings of certain SET command options.

PREServe

The following settings are preserved:

ARBCHAR	SHADOW
CASE	SPAN
CMDLINE	SPILL
CURLINE	SYNONYM
FNAME	TRUNC
LRECL	VARBLANK
NULLS	VERIFY
PREFIX	WRAP
RECFM	ZONE
SCALE	

PRESERVE allows you to save a specific editor configuration for later restoration. You may then modify the SET command options as needed for specific, temporary purposes. Use the RESTORE command to return to the configuration saved by PRESERVE.

PUT

The PUT command writes one or more lines of data from the file being edited into a temporary data buffer or into another file.

PUT [*target* [*filename*]]

With no operands, PUT writes only the current line into a temporary data buffer. Data written to the temporary data buffer may be retrieved using the GET command.

target defines the lines to be written. PUT writes lines beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

filename specifies the name of a file for the output. If ***filename*** does not exist, uni-XEDIT creates the file and writes the lines. If ***filename*** exists, the lines are appended to the end of the file.

SET command options that may affect the results of the PUT command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

`PUT`	inserts only the current line into a temporary data buffer

`PUT *`	writes all lines from the current line to the end of the file into the temporary data buffer

`PUT /xyz/ filea`

writes into “filea” all lines from the current line up to, but not including, the line containing “xyz”

PUTD

The PUTD command writes one or more lines of data from the file being edited into a temporary data buffer or into another file. It also deletes these same lines from the file being edited.

PUTD [*target* [*filename*]]

With no operands, PUTD writes only the current line into a temporary data buffer and deletes only the current line. Data written to the temporary data buffer may be retrieved using the GET or RECOVER command.

target defines the lines to be written and deleted. PUTD writes the lines beginning with the current line up to, but not including, the line identified by ***target*** and deletes the same range of lines. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

filename specifies the name of a file for the output. If ***filename*** does not exist, uni-XEDIT creates the file and writes the lines. If ***filename*** exists, the lines are appended to the end of the file.

SET command options that may affect the results of the PUTD command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

`PUTD`	inserts only the current line into a temporary data buffer and deletes it from the current file
`PUTD *`	writes all lines from the current line to the end of the file into the temporary data buf-fer and deletes this range of lines from the current file
`PUTD /xyz/ filea`	writes into “filea” all lines from the current line up to, but not including, the line con-taining “xyz”; deletes this range of lines from the current file

QUERY

The QUERY command displays on the message line the current settings of a SET command option.

Query ***option*** **[*name*]**

option may be any option of the SET line command. ***name*** is valid only when the option to be queried is SYNONYM.

SET command options that may affect the results of the QUERY command include

MSGLINE
MSGMODE

Examples:

```
QUERY pf5
```

displays the definition for PF5

```
Q synonym advan
```

displays the original command name for the synonym “advan”

QUIT, QQUIT

The QUIT command terminates the edit session if no changes have been made to the file since the last SAVE. The QUIT command is normally assigned to PF3.

QUIT

If you have made changes to the file that have not been saved, the message

```
“577e File has been changed; type QQUIT to quit
anyway”
```

appears.

To discard unsaved changes and terminate the edit session, use the QQUIT command.

QQuit

QQUIT terminates the session regardless of pending changes.

READ

The READ command is intended for use in a macro written in REXX. It places information from the screen onto the REXX program stack in the same order in which it appears on the screen. The macro may then retrieve data from the stack using PULL or PARSE PULL. Conditional macro processing may be based on information retrieved from the screen through the stack.

```
READ    [Cmdline                ]   [Notag]
        [All           [Number]]    [Tag  ]
        [Nochange      [Number]]
```

Cmdline stacks information from the command line only. This is the default.

All stacks all screen lines that have been changed since the user last pressed Enter or a PF key. Information on the command line is stacked last. The file being edited is updated when READ is executed if any changed lines were file lines.

Nochange is similar to **All**. It stacks all screen lines that have been changed since the user last pressed Enter or a PF key. Information on the command line is stacked last. However, the file being edited is **not** updated when READ is executed.

Number specifies that changed lines placed on the stack are prefaced by their file line number.

Notag specifies that no identifying tag be inserted in the stacked lines. This is the default.

Tag inserts an identifying tag at the beginning of each stacked line indicating the origin of the line. When **Tag** is specified and a field is changed, the following tag information is provided:

CMD *string*
: ***string*** is the data from the command line.

CMK *key string*

key is the key, mapped by SET KEYBIND to a line command, that was pressed to terminate READ.

string is the command keybound to that key.

ETK *string*

string is the ENTER key definition.

FIL *n1 n2* [*n3*] *string*

n1 is the screen line number.

n2 is the screen column number of the beginning of the line.

n3 is the file line number. It is returned only when the Number operand of READ is used.

string is the data from the changed line.

PFK *n string*

n is the number of the PF key pressed to terminate READ.

string is the PF key definition.

PRF *n1 n2* [*n3*] *string*

n1 is the screen line number.

n2 is the screen column number of the prefix area.

n3 is the file line number. It is returned only when the Number operand of READ is used.

string is the data from the prefix area.

RES *n1 n2 string*

n1 is the screen line number.

n2 is the screen column number of the beginning of the reserved area.

string is the data from the reserved line.

Examples:

```
read all number
```

places on the REXX program stack data from all lines that have been changed since the user last pressed Enter or a PF key; at the same time, the data in the file is updated.

```
read n t
```

places on the REXX program stack data from all lines that have been changed since the user last pressed Enter or a PF key; the data in the file is **not** changed; each line stacked is tagged to indicate the origin of the line.

The section entitled “User Applications Based on uni-XEDIT” in ***Chapter 6: Edit Macros*** contains an example of the use of READ. In addition, the uni-XEDIT Sample Library, distributed with the editor, includes an example of the use of READ in a uni-XEDIT based application. The adbook application uses two edit macros in which processing is based on the results of a READ command.

RECOVER

The RECOVER command restores the line or lines previously removed by the DELETE or PUTD line command or the D prefix command. Recovered lines are inserted immediately above the current line. The last recovered line becomes the new current line.

RECover [*n*]

Without operands, RECOVER restores only the last deleted line. ***n*** specifies the number of lines to recover. ***n*** may be a number, or it may be * to recover all deleted lines remaining in the buffer.

Examples:

`RECOVER`	recovers the last line deleted and places it above the current line. The inserted line becomes the new current line.
`REC 6`	recovers the last six lines deleted. The sixth line recovered becomes the new current line.
`REC *`	recovers all deleted lines. The last line recovered becomes the new current line.

REFRESH

The REFRESH command causes the current screen display to be sent to the terminal in its entirety.

REFRESH

Refresh is used when something causes the uni-XEDIT screen display to be damaged, as when an operating system message is written on the screen.

In normal operation, uni-XEDIT does not write the entire screen, but only those areas that require updates based on the most recent operation. Therefore, a part of the screen displaying an operating system message may appear to overlay the normal uni-XEDIT screen display. The REFRESH command causes the entire screen to be redisplayed, removing any unwanted messages on the screen.

REPEAT

The REPEAT command repositions the current line and re-executes the last line command entered.

REPEat [*target*]

With no operands, REPEAT advances the current line one line and re-executes the last command once.

target defines the number of times that the command is to be repeated. When ***target*** is specified, REPEAT computes the number of lines between the current line and the line identified by ***target***. That number controls how many times the command is repeated. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the REPEAT command include

RANGE

Examples:

`REPEAT`	moves the current line forward one line and repeats the last command
`REPE 3`	moves the current line forward and repeats the last command three times
`REPE -/xyz/`	moves the current line backward; computes the number of lines between the current line and the first previous line that contains “xyz”; repeats the last command that many times

REPLACE

The REPLACE command replaces the contents of the current line with the text specified.

Replace [*text*]

With no operands, REPLACE replaces the current line with a blank line.

The ***text*** operand specifies the new contents for the current line.

When SET SPILL OFF is in effect, characters in ***text*** that would extend beyond the truncation column are truncated. When SET SPILL ON or SET SPILL WORD is in effect, characters that would extend beyond the truncation column are inserted as one or more new lines in the file.

SET command options that may affect the results of the REPLACE command include

- CASE
- SPILL
- TRUNC

Example:

`REPLACE` replaces the current line with a blank line

`R **application one**`
replaces the current line with the text "**application one**" beginning in column 1

RESET

The RESET command removes pending prefix commands.

RESET

Block prefix commands and prefix commands that require a destination prefix remain pending until the necessary matching command is entered. The RESET command removes all pending prefix commands.

Prefix commands that do not require a matching prefix are automatically executed whenever you press Enter. Thus, they cannot be removed by RESET. To remove such a prefix, type over it with another prefix or a blank, or erase it with the DELETE or ERASE-EOF hardware edit key.

RESTORE

The RESTORE command restores the editor configuration to the state that existed at the time of the last PRESERVE command.

RESTore

The following settings are restored:

ARBCHAR	SHADOW
CASE	SPAN
CMDLINE	SPILL
CURLINE	SYNONYM
FNAME	TRUNC
LRECL	VARBLANK
NULLS	VERIFY
PREFIX	WRAP
RECFM	ZONE
SCALE	

RGTLEFT

The RGTLEFT command shifts the edit screen display right or left one or more columns to facilitate viewing data beyond the current display. The RGTLEFT command, without operands, is normally assigned to PF10.

RGTLEFT [*n*]

With no operands, RGTLEFT shifts the display right or left to display data that is currently off the screen. The display is never shifted more that seventy-five percent of the screen width.

n specifies the number of columns to move the display in the direction of the off-screen data. When ***n*** is specified as zero, RGTLEFT positions the display so that column 1 of the data is in the leftmost display position.

When you first enter the editor, unseen data is to the right of the current screen display. Pressing PF10 scrolls right to display this data. Pressing PF10 again scrolls left to the original display.

RIGHT

The RIGHT command shifts the edit screen display one or more columns to the right, allowing you to view data that is to the right of the current display.

RIght [*n*]

With no operands, RIGHT shifts the display by one column. ***n*** specifies the number of columns to shift right. When ***n*** is specified as zero, RIGHT positions the display so that column 1 of the data is in the left-most display position.

Examples:

`RIGHT 10`	moves the display right 10 columns
`RI`	moves the display right 1 column
`RI 0`	displays the left-most screen of data, beginning with column 1

SAVE, SSAVE

The SAVE command writes the current file to disk and remains in the edit session.

SAVE [*filename*]

When ***filename*** is omitted, SAVE writes the file to disk using the filename displayed on the File-id line. Use the ***filename*** operand to save the file under a different name.

When you enter a SAVE command, uni-XEDIT checks for two conditions that would inhibit saving the file:

- another user has changed the file since your edit session started (for saving under the current name)
- a file by the name ***filename*** already exists (for saving under a different name)

In either case, uni-XEDIT displays an appropriate message, does not save the file, and remains in the edit session. In the first instance, the message is

```
1030e File filename has been modified by another
user; use FFILE/SSAVE
```

In the second instance, the message is

```
594e File filename already exists; use FFILE/SSAVE
```

To save the file without regard to the current status of the disk file, use the SSAVE command.

SSave [*filename*]

SET command options that may affect the results of the SAVE or SSAVE command include

FNAME
LRECL

Multiple Edit Protection

The action of the SAVE command provides some protection for multiple users editing the same file. In this context, multiple users may be any of the following:

- two or more userids editing the same file with uni-XEDIT
- two or more userids editing the same file with uni-XEDIT and possibly other editors
- one userid editing the same file in two or more windows with uni-XEDIT and possibly other editors

Because of the way this protection is implemented, it is effective across all NFS links as well as on the current workstation.

When you bring a file into uni-XEDIT, the editor captures the “last modified” date/time stamp of the file. When you later enter a SAVE command, the editor again checks the “last modified” date/time stamp of the disk file before performing the save. If the current date/time stamp does not match the original date/time, message number 1030e appears.

You then have the option to save the file under another name. This action protects any changes that may have been made by another user. SSAVE bypasses the date/time stamp checking and overwrites the existing file. This action destroys any changes that may have been made by another user.

File protection based on the date/time stamp is only invoked when the name under which you are saving the file is the same as the name under which you loaded it. If you specify a different filename on the SAVE command or if you use SET FNAME to change the default name for a SAVE command, the usual tests for existence of a file by the new name apply. For these purposes the filenames “junk” and “./junk” are not equivalent even though “junk” resides in the current directory.

File-System-Full Protection

uni-XEDIT also provides protection against loss of data in the event of file-system-full conditions. Two alternative protection mechanisms are available as well as an option to disable this protection.

During a FILE or SAVE operation, uni-XEDIT protects data by copying the current session to a temporary file, removing the original file, and replacing it with the temporary copy. In the event of a file-system-full condition, a message appears indicating the condition and the location of the preserved copy. In the event that the temporary file cannot be created, you may use the FFILE or SSAVE command to bypass the protection.

The cnvironment variable XETEMP may be used to control the location of the temporary file or to bypass the protection mechanism entirely.

- When XETEMP is not defined, uni-XEDIT writes the temporary file in the same directory as the original file. This is the default behavior.
- When XETEMP is set to a fully qualified directory name, uni-XEDIT writes the temporary file to this directory.
- When XETEMP is set to “NOTEMP”, uni-XEDIT disables file protection completely. No temporary file is created and all attempts to save the file write directly to the original file. (NOTEMP must be specified in uppercase and cannot be abbreviated.)

The most complete protection is provided when XETEMP identifies a directory with ample free space in a file system other than the one where user files typically reside. This directory must have write permission for all users and ample space to save the temporary copies of your largest files.

Excellent protection is also provided through the default mechanism, when XETEMP is not set. If a file-system-full condition prevents creation of the temporary file, you may save the file to a different file system by specifying a full path name on the FILE or SAVE command. Alternatively, you may use FFILE or SSAVE to bypass the protection mechanism.

The greatest exposure occurs when XETEMP is set to “NOTEMP”. This disables protection completely and leaves you vulnerable to loss of data. Any attempt to save a file writes directly over the original file. If a file-system-full condition occurs, the output file is truncated at that point and the updated file exists **only** in the current edit session.

Examples:

`SAVE`	writes the edited file to disk using the same name displayed on the File-id line, overlaying the existing file with the same name
`SSAVE`	writes the edited file to disk using the same name displayed on the File-ID line, overlaying the existing file with the same name regardless of whether it has been modified by another user
`SAVE acctbal`	writes the edited file to the disk file “acctbal” but does not overwrite “acctbal” if it already exists
`SS acctbal`	writes the edited file to “acctbal” even if it already exists

SCHANGE

The SCHANGE command changes the PF key assigned to the selective change operation. PF5 is normally defined as SCHANGE 6.

SCHANGE ***keynumber***

keynumber is the PF key number to perform the selective change.

The documentation of the CHANGE command in this chapter contains a complete description of the selective change feature.

SET

Options of the SET command control configuration parameters for the current editing environment.

[SET] ***option***

The SET command itself is optional. When you specify only ***option*** on the command line, SET is assumed.

option may be any one of the following:

You may SET an ***option*** interactively as needed. Frequently used set options may be included in your uni-XEDIT profile. ***Chapter 5: SET Command Options*** provides complete information, including examples, about each option of the SET command.

To determine interactively the current setting of a SET option, use the QUERY command. To determine this information in a macro, use the EXTRACT command.

SHELL

The SHELL command provides access to the Unix shell for command execution. The specific shell used is determined by the current setting of the SHELL environment variable.

SHELL [*string*]

With no operands, SHELL exits to the Unix shell for execution of one or more shell commands. In graphical mode, a new terminal window appears for shell com-

ALT	**MSGMODE**	**SCREEN**
ARBCHAR	**NONDISP**	**SHADOW**
AUTOSAVE	**NULLS**	**SHELLTERM**
CASE	**NUMBER**	**SPAN**
CMDLINE	**PENDING**	**SPILL**
CTLCHAR	**PF**	**STAY**
CURLINE	**POINT**	**SYNCSCROLL**
DISPLAY	**PREFIX**	**SYNONYM**
ENTER	**RANGE**	**TABLINE**
FNAME	**RECFM**	**TABS**
KEYBIND	**RESERVED**	**TRUNC**
LINEND	**SELECT**	**VARBLANK**
LRECL	**SELMODE**	**VERIFY**
MASK	**SCALE**	**WRAP**
MSGLINE	**SCOPE**	**ZONE**

mand execution. To return to the uni-XEDIT session, you may either type “exit” at the Unix prompt or close the terminal window. In character mode, shell command I/O occurs at the bottom of the current window, which scrolls upward as commands are executed. To return to the uni-XEDIT session, type “exit” at the Unix prompt. The message

```
    Press ENTER to return to the editor
```

appears. Press Enter to resume the edit session.

string is a valid Unix command that is passed to the shell for execution. It is a single command that is executed immediately. The message

```
    Press ENTER to return to the editor
```

appears after the output of the Unix command.

The SHELL command is synonymous with the CMS and ! commands.

SET command options that may affect the results of the SHELL command include

SHELLTERM (graphical mode only)

Example:

```
SHELL ls
```

displays a file listing of the current directory followed by the message “Press ENTER to return to the editor”

SHIFT

The SHIFT command moves data one or more columns to the left or right on one or more lines. Data shifted to the left of the *zone-1* column is lost. Data shifted to the right of the truncation column may also be lost.

SHift	**Left**	**[*cols*]**	**[*target*]**
	Right		

When ***cols*** is omitted, data is shifted one column in the direction specified. ***cols*** specifies the number of columns to shift.

When ***target*** is omitted, data is shifted on the current line only. ***target*** defines the number of lines to shift. Data is shifted beginning on the current line up to, but not incuding, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

The operation of the SHIFT command is different from that of RIGHT, LEFT, RGTLEFT, or SET VERIFY. These commands shift the display screen to facilitate viewing different areas of the display. The SHIFT command actually relocates data within the file.

When SET SPILL ON or SET SPILL WORD is in effect, characters or words shifted right beyond the truncation column are inserted in the file as one or more new lines. SET SPILL affects SHIFT RIGHT **only**. It is not possible to preserve data shifted left beyond the *zone-1* column.

SET command options that may affect the results of the SHIFT command include

ARBCHAR
CASE
RANGE
SPAN
SPILL
TRUNC
VARBLANK
ZONE

Examples:

```
SHIFT LEFT
```

shifts data one column to the left on the current line only. If the data being shifted is in the *zone-1* column, it is lost and cannot be recovered.

```
SH L 4 *
```

shifts data left 4 columns on all lines from the current line to the end of the file. If the data being shifted begins in the *zone-1* column, it is lost and cannot be recovered.

```
SH R 3 /abc/
```

shifts data right 3 columns on all lines from the current line up to, but not including, the first line that contains “abc”

SORT

The SORT command reorders lines in the file you are editing. SORT always reorders complete records.

SORT *target* **[A]** ***sort-field-1*** **[... *sort-field-n*]**
[D]

target defines the number of lines to be sorted. Lines are reordered beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

The second operand is optional and specifies the sort order. When this operand is omitted, the default Ascending (**A**) order is used. Sort order is determined by the hexadecimal representation of the characters in the data.

Lines are reordered based on the contents of one or more sort-fields. A ***sort-field*** consists of a pair of column numbers that specify the beginning and end of the field, respectively. You may specify any number of sort-fields. Each sort field is a secondary key to the one that precedes it as the command was entered.

SET command options that may affect the results of the SORT command include

ARBCHAR
CASE
RANGE
SPAN
TRUNC
VARBLANK
ZONE

Examples:

```
SORT * 1 21 42 46
```

reorders all lines from the current line to the end of the file in ascending order using columns 1-21 as the primary sort-field and columns 42-46 as the secondary sort-field

```
SORT /xyz/ 1 21 42 46
```

reorders all lines from the current line up to the first line that contains “xyz” in ascending order using columns 1-21 as the primary sort-field and columns 42-46 as the secondary sort-field. The line containing “xyz” is not included in the sort.

```
SORT 5 D 1 50
```

reorders the current line and the following 4 lines in descending order using columns 1-50 as the only sort-field

SOS

SOS commands provide a set of high-level functions to be used within macros or assigned to PF keys.

SOS *option*

option may be any of the following keywords:

Alarm
generates an alarm the next time the screen is refreshed

CLEAR
clears the logical screen

LINEAdd
adds a blank line after the line where the cursor is currently positioned

LINEDel
deletes the line where the cursor is currently positioned

NUlls **[ON]**
[OFF]
specifies whether trailing blanks on each line are displayed as spaces (hexadecimal 20) or as nulls (hexadecimal 00). Currently, this option has no effect.

PF *n*
causes any data or commands assigned to PF***n*** to be placed onto the program stack

POP
removes the top entry from the cursor stack and places the cursor there. If the resulting cursor position would be outside the logical screen, then the cursor is positioned in the upper left position of the current logical screen.

PUsh

saves the current cursor position by placing it on the top of the cursor stack in last-in-first-out (LIFO) order

TABB [*n*]

moves the cursor to the previous tab position relative to the current cursor position. ***n*** is a positive number that defines the number of tab positions to move. If ***n*** is omitted, the default is 1.

TABCmd

moves the cursor to the command line of the current logical screen

TABCMDB

moves the cursor to the command line of the previous logical screen

TABCMDF

moves the cursor to the command line of the next logical screen

TABF [*n*]

moves the cursor to the next tab position relative to the current cursor position. ***n*** is a positive number that defines the number of tab positions to move. If ***n*** is omitted, the default is 1.

SPLTJOIN

The SPLTJOIN command splits a line into two lines when the cursor is located on or before the last non-blank character of a line. SPLTJOIN joins two lines into a single line when the cursor is located after the last non-blank character in a line. The SPLTJOIN command is normally assigned to PF11.

SPLTJOIN

If SPLTJOIN adds data that would extend beyond the truncation column, a new line is started at that point. No data is ever truncated.

STATUS

The STATUS command returns the current settings of the SET command options.

STATus [*filename*]

With no operands, STATUS displays its output outside of the normal editor screen configuration. The output is followed by the message

```
Press ENTER to return to the editor
```

filename specifies the name of the file in which the output from STATUS is captured. The output for each SET option is in the form

`'command set` *`option settings`*`'`

so that the file is suitable for use as a uni-XEDIT profile.

TOP

The TOP command scrolls the screen display to the beginning of the file so that the “Top of File” or “Top of Range” line becomes the current line.

TOP

When SET RANGE has been used to redefine the top line of the file, TOP scrolls to “Top of Range” line.

UP

The UP command advances the current line toward the top of the file.

Up [*n*]

With no operands, UP moves the current line up one line. ***n*** specifies the number of lines to move. ***n*** may be a number, or it may be * to advance up to the top of the file.

SET command options that may affect the results of the UP command include

RANGE

Examples:

`UP 5`	moves the current line up five lines
`UP *`	moves the current line to the “Top of File” or “Top of Range” line

UPPERCAS

The UPPERCAS command converts lowercase letters to uppercase letters on one or more lines in the file.

UPPercas [*target*]

With no operands, UPPERCAS converts only the current line.

target defines the range of lines to convert. UPPERCAS converts lines beginning with the current line up to, but not including, the line identified by ***target***. ***target*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

SET command options that may affect the results of the UPPERCAS command include

ARBCHAR
CASE
RANGE
STAY
TRUNC
ZONE

Examples:

`UPPERCAS`

converts only the current line to uppercase

`UPP 5` converts the current line and the following 4 lines to uppercase

`UPP -/xyz/`

converts to uppercase the current line and all preceding lines in the file up to, but not including, the first line containing “xyz”

XEDIT

The XEDIT command performs three functions:

- adds a file to the edit ring
- cycles through the files in the edit ring
- moves immediately to the specified file in the edit ring

Xedit [*filename*] [-Prof *profname*]
[-NOProf]

With no operands, XEDIT displays the next file in the edit ring.

filename specifies the name of the file to display. If this file is not already in the edit ring, XEDIT adds it to the ring. If ***filename*** is already in the ring, XEDIT immediately moves the display to ***filename***.

The **-Prof** operand allows you to specify a different profile for ***filename*** than the profile currently in use for other files in the ring. ***profname*** is the name of the profile to be used with this file.

-NOProf brings filename into the edit ring with no profile in effect.

The **-Prof** and **-NOProf** operands of the XEDIT line command function exactly like their counterpart options of the editor startup command, “xe”. The same rules for locating ***profname*** apply as well. ***Chapter 2: Editor Operation*** includes these details.

Examples:

`X`	displays the next file in the ring
`X dat1`	if “dat1" is already in the ring, displays it immediately. If ”dat1" is not in the ring but exists as a disk file, adds it to the ring and displays it. If “dat1" is not in the ring, and

does not exist as a disk file, adds it to the ring as a new file and displays it.

`x dat1 -p datprof`

If “dat1" is not in the ring but exists as a disk file, adds it to the ring and displays it using the commands in the file datprof.xedit to configure the editor. If ”dat1" is not in the ring and does not exist as a disk file, adds it to the ring as a new file and displays it; the commands in datprof.xedit are used to establish the editor configuration for the new file. If “dat1" is already in the ring, the editor displays it and the -p operand is ignored.

= (Equal Sign)

The = command re-executes the last command entered. It is normally assigned to PF9.

= [*command*]

With no operands, = simply re-executes the last command. This command could have been originally executed from the command line or from a PF key. A series of equal signs (===) re-executes the last command multiple times.

The ***command*** operand specifies a particular command to be executed before repeating the last command. When ***command*** is specified, = first executes ***command*** and then executes the previous command.

Example:

Last command:	`clocate /xyz/`
Next command:	`=`
Results:	the column pointer moves to the next occurrence of “xyz”
Next command:	`= TOP`
Results:	the current line moves to the top of the file and the column pointer then moves to the first occurrence of “xyz” after the current column pointer position

? (Question Mark)

The ? command redisplays on the command line the last command executed. The ? command is normally assigned to PF6.

?

Previous commands are retained in a buffer and are retrieved in a "last-in-first-out" (LIFO) order. To redisplay a command from the buffer that is not the immediately previous command, use a series of question marks. The number of question marks corresponds to the number in the buffer of the command you want to retrieve. If there are no commands remaining in the buffer, ? displays the command you used to initiate this edit session.

Examples:

`?`	redisplay the last command entered on the command line
`???`	redisplay the third-from-last command entered

! (Exclamation Point)

The ! command provides access to the Unix shell for command execution. The specific shell used is determined by the current setting of the SHELL environment variable.

![*string*]

With no operands, ! exits to the Unix shell for execution of one or more shell commands. In graphical mode, a new terminal window appears for shell command execution. To return to the uni-XEDIT session, you may either type "exit" at the Unix prompt or close the terminal window. In character mode, shell command I/O occurs at the bottom of with the current window, which scrolls upward as commands are executed. To return to the uni-XEDIT session, type "exit" at the Unix prompt. Then press Enter to resume the edit session.

string is a valid Unix command that is passed to the shell for execution. It is a single command that is executed immediately. Note that there is no delimiter space between the ! command and its ***string*** operand. The message

```
    Press ENTER to return to the editor
```

appears after the output of the Unix command.

The ! command is synonymous with the CMS and SHELL commands.

SET command options that may affect the results of the ! command include

SHELLTERM (graphical mode only)

Example:

`!ls`	displays a file listing of the current directory followed by the message "Press ENTER to return to the editor"

& (Ampersand)

The & command is used as a modifier of other line commands. It cannot be used as an independent command.

& command

command may be any uni-XEDIT line command.

The editor executes ***command*** and redisplays it on the command line, along with the & modifier. This allows you to re-execute ***command*** easily or to modify it before re-executing it.

You may not set a synonym for the & command.

MSGMODE must be ON for & to function.

Examples:

`& down 4`	moves the current line down 4 lines and residplays `& down 4` on the command line; you may then modify the command for different results or re-execute it exactly as it appears

Chapter 5: SET Command Options

The SET line command supports an array of options through which you control the editing environment by

- re-configuring the screen display
- defining PF keys
- varying the column- and/or line-range for editor operations
- defining keyboard mappings
- controlling case sensitivity for string operands
- assigning symbolic names to lines of data
- establishing a frequency for automatic saves
- and controlling a variety of other session parameters

The options are:

ALT	**MSGLINE**	**SCREEN**
ARBCHAR	**MSGMODE**	**SHADOW**
AUTOSAVE	**NONDISP**	**SHELLTERM**
CASE	**NULLS**	**SPAN**
CMDLINE	**NUMBER**	**SPILL**
COLOR	**PENDING**	**STAY**
CTLCHAR	**PF**	**SYNCSCROLL**
CURLINE	**POINT**	**SYNONYM**
DISPLAY	**PREFIX**	**TABLINE**
ENTER	**RANGE**	**TABS**
FNAME	**RECFM**	**TRUNC**
HEX	**RESERVED**	**VARBLANK**
KEYBIND	**SELECT**	**VERIFY**
LINEND	**SELMODE**	**WRAP**
LRECL	**SCALE**	**ZONE**
MASK	**SCOPE**	

The syntax of the SET command is

[SET] ***option***

The SET command itself is optional. When you specify only ***option*** on the command line, SET is assumed.

You may SET an option interactively as needed. Frequently used options may be included in your uni-XEDIT profile. The QUERY and EXTRACT line commands allow you to determine the current setting of an option interactively or in a macro, respectively.

The sections which follow provide complete details on the use of each SET command option, including the default settings. Also included are examples and a list of commands affected by each option.

ALT

The ALT option controls the setting of one or both of the alteration counters and is intended for use in macros. uni-XEDIT provides two alteration counters that contain a count of the number of changes made to a file:

- Autosave Counter – the number of changes made since the last AUTOSAVE occurred
- Save Counter – the number of changes made since the last SAVE command

A change is defined as any alteration to the file resulting from execution of a prefix command, a line command, or data modification.

[SET] ALT *n* [*p*]

Initial default: 0 0

n specifies the value to be assigned to the Autosave Counter and must be a positive integer.

p specifies the value to be assigned to the Save Counter and must be a positive integer. When ***p*** is omitted, the Save Counter remains unchanged.

ARBCHAR

The ARBCHAR option controls the use of arbitrary characters in a string target or in the ***string-1*** operand of the CHANGE command. It also defines the special character to be used.

[SET]	**ARBchar**	**ON**	**[*character*]**
		OFF	

Initial default: ARBCHAR OFF $

ON enables the use of arbitrary characters.

The ***character*** operand defines the specific arbitrary character. It must be a special character such as $, ?, %, or @. If omitted, the most recently specified arbitrary character is used. To use "#" as an arbitrary character, you must either SET LINEND OFF or redefine the logical line-end character.

OFF disables the use of arbitrary characters in string operands. It does not affect the selection of the specific arbitrary character.

When the arbitrary character appears in a string operand, it matches any number of characters in the data. Used alone, it matches the entire line of data. Used in conjunction with literals, it matches any number of characters before, after, or between the literals.

SET ARBCHAR may affect the results of any command that supports a string target operand. It may also affect the results of the CHANGE command.

Examples:

`ARB ON ?` enables the use of arbitrary characters and defines the character as ?. In a subsequent command such as `L /hot?dog/`, "hot?dog" matches any of the following:

hotdog
hot-dog
hot diggity dog
hot chili and cheese dog

AUTOSAVE

The AUTOSAVE option controls automatic interim saves of the data being edited.

[SET] AUtosave ***frequency***
OFF

Initial default: AUTOSAVE OFF

frequency specifies the number of changes allowed between saves. A change is defined as any alteration to the file brought about by the execution of a prefix command, line command, or data entry. The "alt=" count on the File-id line of the uni-XEDIT screen registers the number of changes since the last save. The "alt=" count is reset to zero after each autosave.

The interim version of the data is stored in the same directory as the file being edited. The name of the backup file is in the form

1***nnnnn***.AUTOSAVE

where *nnnnn* begins as "00001" and is incremented for each file in the edit ring. The new backup file replaces the previous one each time an autosave occurs. The message

"510i Autosaved as 1***nnnnnn***.AUTOSAVE"

appears in conjunction with each autosave.

OFF turns the autosave feature off.

Examples:

`SET AUTOSAVE 20`
sets the autosave frequency to 20. A backup file is saved whenever the "alt=" count reaches 20.

`AU off` turns the autosave feature off

CASE

The CASE option controls case sensitivity for string operands and for data entered into the file.

[SET]	**CASE**	**Mixed**	**[Respect]**
		Uppercase	**[Ignore]**

Initial default: CASE MIXED RESPECT

When **Mixed** is in effect, data is placed in the file exactly as it was entered on a data line or as the string operand of a command. String targets are also retained in the case in which they were typed.

When **Uppercase** is in effect, data entered on a data line or as the string operand of a command is automatically converted to uppercase. String targets are also converted automatically.

The Respect/Ignore operand controls case sensitivity for target searches and for the FIND and FINDUP commands. **Respect** indicates that searches are case sensitive. **Ignore** indicates that searches are case insensitive.

SET CASE may affect the results of any command that supports a string operand, including string targets.

Examples:

`SET CASE UPPER RESPECT`

all data entered in the file is automatically converted to uppercase. Target searches and the FIND and FINDUP commands remain case sensitive.

`CASE M I` data entered into the file is preserved in the case in which it was typed. Target searches and the FIND and FINDUP commands are case insensitive.

CMDLINE

The CMDLINE option controls the position and display of the command line on your edit screen.

```
[SET]  CMDline    [On]
                  [OFf]
                  [Top]
                  [Bottom]
```

Initial default: CMDLINE BOTTOM

Specifying **On** positions the command line after the last line of data and before the product version indicator. This is equivalent to specifying **Bottom**. Specifying **Top** positions the command line immediately after the File-id line. Specifying **OFf** removes the command line from the screen.

If you specify **OFf**, the only commands available are those assigned to PF keys. It is therefore recommended that you define a PF key as “SET CMDLINE ON” before turning the command line off.

COLOR

The COLOR option associates specific colors with named areas of the edit screen.

[SET] COLOR <*area*> *color* [*exthi*] [High]|[Nohigh]
<*>

* indicates all of the areas below

Initial Defaults:

FIELD	COLOR	EXTHI	HIGH/NOHIGH
ARROW	DEFAULT	NONE	HIGH
CMDLINE	DEFAULT	NONE	NOHIGH
CURLINE	DEFAULT	NONE	HIGH
FILEAREA	DEFAULT	NONE	NOHIGH
IDLINE	DEFAULT	NONE	HIGH
MSGLINE	RED	NONE	HIGH
PENDING	DEFAULT	NONE	HIGH
PREFIX	DEFAULT	NONE	NOHIGH
SCALE	DEFAULT	NONE	HIGH
SHADOW	DEFAULT	NONE	NOHIGH
STATAREA	DEFAULT	NONE	HIGH
TABLINE	DEFAULT	NONE	HIGH
TOFEOF	DEFAULT	NONE	NOHIGH

area can be any of the following:

arrow	the arrow preceding the command line**.**
cmdline	the line where commands are entered.
curline	the current line within the file area.
filearea	the file data area, excluding the current line.
idline	the file identifier, line one of the edit screen.
msgline	the screen area for message display.
pending	the pending subcommand as entered.
prefix	the five character area to the right of lines in the filearea.

scale	the scale line.
shadow	lines resulting from selective editing.
statarea	status area in lower right-hand corner.
tabline	the line which displays current tab settings.
tofeof	the top of file and end of file indicator lines.

color can be any of: blue, red, pink, green, turquoise, yellow, white, default.

exthi can be any of: blink, revvideo, underline, none.

High|Nohigh refers to high or normal intensity.

CTLCHAR

The CTLCHAR option defines characters used in control character sequences and the display characteristics associated with a control character. It is intended for use in a macro that uses reserved lines to display special fields. A control character sequence is composed of an escape character and a control character and specifies special display characteristics for the field that immediately follows it. Any alphanumeric or special character may be defined as either an escape character or a control character. Escape and control characters are effective for all files in the ring and across multiple screens. Although the control character sequence does not appear in the display, it occupies one display position on the screen.

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

Initial defaults: OFF

OFF (with no additional operands) resets all defined escape and control characters to normal text characters.

char specifies the character to be defined. The operand(s) that follow ***char*** specify its purpose and the display characteristics associated with it. To use "#" as **char**, you must first SET LINEND OFF or redefine the logical line-end character.

Escape defines ***char*** as the escape character. Each control character sequence must begin with an escape character. Only one escape character is recognized in each macro. If you define more than one character as the escape character, only the last definition has any effect.

Protect defines ***char*** as a control character that identifies a protected field. Users may not type data in protected fields. Optional display characteristics that may be associated with ***char*** control the appearance of the data on the screen.

Noprotect defines ***char*** as a control character that identifies an unprotected field. Users may type into unprotected fields. Optional display characteristics that may be associated with ***char*** control the appearance of the data on the screen.

High specifies that data in the field is highlighted. Although highlighting is not supported in this release of uni-XEDIT, the syntax is supported for macro compatibility.

Nohigh specifies that data in the field is not highlighted. This is the default.

Invisible specifies that data in the field is not displayed.

OFF resets ***char*** as a normal text character.

Examples:

```
set ctlchar % escape
```

defines % as the escape character which begins a control character sequence

```
ctl @ noprotect
```

defines @ as a control character that denotes an unprotected field

```
ctl a p i
```

defines “a” as a control character that denotes a protected field in which the data is not displayed

```
ctl off
```

resets all escape and control characters to normal text characters `ctl @ off`

resets the @ character to a normal text character

The section entitled “User Applications Based on uni-XEDIT” in ***Chapter 6: Edit Macros*** contains an example of the use of SET CTLCHAR. In addition, The uni-XEDIT Sample Library, distributed with the editor, includes an example of the use of control characters and reserved lines in a uni-XEDIT based application. The adbook application uses two edit macros in which control characters are defined and used to configure reserved lines.

CURLINE

The CURLINE option controls the position of the current line on the screen.

[SET] CURLine ON *pos*

Initial default: CURLINE ON M

pos designates the location on the screen where the current line should appear. ***pos*** may be any of the following:

M	middle of the screen
M+*n*	*n* lines below the middle of the screen
M-*n*	*n* lines above the middle of the screen
n	*n* lines below the top of the screen
+*n*	same as *n*
-*n*	*n* lines above the bottom of the screen

Examples:

```
SET CURLINE ON M-5
```

places the current line 5 lines above the middle of the screen

```
CURL ON 7
```

places the current line on the seventh line of the screen

DISPLAY

The DISPLAY option controls the lines that appear on the screen. It is used in conjunction with the SELECT, SCOPE, and SHADOW options to include or exclude lines from the display.

[SET] DISPlay *n1* [*n2*]

Initial Default: DISPLAY 0 0

n1 is the selection level of lines to be shown on the screen. If ***n2*** is omitted, only lines with a selection level of ***n1*** are displayed.

n2 is a secondary selection of lines to be shown. If ***n2*** is specified, ***n1*** and ***n2*** define a range of selection levels to display. ***n2*** must be greater than or equal to ***n1***. If ***n2*** is specified as "*", all lines with selection levels greater than ***n1*** are displayed.

ENTER

The ENTER option changes the function of the Enter key.

[SET] ENTer [BEFORE] [*string*]
[AFTER]
[ONLY]
[IGNORE]

Initial default:

ENTER IGNORE CURSOR CMDLINE 1 PRIORITY 30

string may be any uni-XEDIT line command. When specified, ***string*** defines a function to be performed by the Enter key in addition to any pending command line or prefix input. When ***string*** is omitted, the Enter key executes only pending input.

BEFORE specifies that ***string*** is executed prior to pending line or prefix commands.

AFTER specifies that ***string*** is executed after pending line or prefix commands.

ONLY specifies that only ***string*** is executed. Pending command line input is discarded. Pending prefix commands are executed. **ONLY** is the default if ***string*** begins with '?' or '='.

IGNORE specifies that ***string*** is not executed when command line input is present.

SET ENTER may affect the results of any line command or prefix command.

Examples:

```
ENTER BEFORE DOWN 3
```

When you press Enter, the current line moves down 3 lines. If any line commands or prefix commands are pending, they are executed after the current line is repositioned.

FNAME

The FNAME option changes the name of the file you are currently editing.

[SET] FName ***filename***

Initial default: current name of the file

filename may be any valid Unix filename. The new name appears on the File-id line. Subsequent FILE or SAVE commands use the new name.

SET FNAME may affect the results of the following line commands:

FILE
FFILE
SAVE
SSAVE

HEX

When ON, allows subcommand string operands to be specified in hexadecimal notation.

[SET] HEX ON|OFF

Initial Default: OFF

KEYBIND

The KEYBIND option allows you to define custom keyboard mappings for any terminal/workstation combination.

The section “Keyboard Mapping” in ***Chapter 2: Editor Operation*** contains detailed information on terminal support and keyboard mapping by uni-XEDIT, including a discussion of the use of the Unix System V curses library and terminfo database to automatically recognize certain keys. This section describes the use of SET KEYBIND to create custom mappings.

The uni-XEDIT keyboard test and keybind maintenance utility, uni-KEY, automates this process for you. ***Appendix C: uni-KEY Reference*** contains complete documentation of the use of uni-KEY.

You may use SET KEYBIND in two ways:

- to map keys on your keyboard that are not automatically recognized by curses
- to override the curses mapping of keys for further customization of your keyboard

In either case, you may map a key to a uni-XEDIT keyboard function or a uni-XEDIT command.

[SET] KEYBind [*mode*] *seq* KEY *keyname*
**   *c-name* COMMAND *cmd***

Use the ***mode*** operand to define mode-specific keybinds. ***mode*** must be specified as “G” for graphical mode or “C” for character mode. When ***mode*** is “G”, the keybind is effective for both graphical and character mode edit sessions. When ***mode*** is “C”, the keybind is effective only for character mode sessions. If ***mode*** is omitted, the default is “G”.

Use the ***seq*** operand to map keys that are not automatically recognized by curses. ***seq*** is the keycode sequence transmitted when you press a key on the keyboard. ***seq*** may include Control, Escape, or other special codes that are normally non-printing characters.

SET KEYBIND uses a standard character representation for each of these non-printing sequences. Use the following table to specify the indicated parts of the code sequence:

^	Control
\e or \E	Escape (0x1b)
\x or \X	Hexadecimal
\[0-3]	Octal
\l or \n	Newline (0x0a)
\r	Return (0x0d)
\t	Tab (0x09)
\b	Backspace (0x08)
\f	Formfeed (0x0c)
\s	Space (0x20)
\^	Caret
\/	Slash
\,	Comma
\:	Colon

Examples of keycode sequences:

`\e1`	escape 1
`^x`	control x
`\x0d`	hexadecimal value `0d'

Note that ***seq* must** begin with a caret (^) or a backslash (\) character.

Use the ***c-name*** operand to override the curses mapping of a key. ***c-name*** is the name generated internally by curses for each key on the keyboard. curses key names are in the form

KEY_<*function*>

where ***function*** is the function performed by this key. You may enter ***c-name*** in upper or lower case.

Many curses key names (such as KEY_TAB) are common across all implementations of Unix. Other names (such as KEY_ACTION for the IBM RS/6000 running AIX) are specific to a particular vendor's implementation. Vendor-specific curses key names are often associated with keys that are unique to the keyboards manufactured by that vendor.

Many of the curses key names are automatically processed by uni-XEDIT for specific keyboard functions. The following table provides a list of commonly available curses key names and the uni-XEDIT function automatically mapped to that name:

curses-name	**uni-XEDIT Function**
KEY_BACKSPACE	backspace
KEY_BTAB	backtab
KEY_DC	delete character
KEY_DOWN	cursor down
KEY_END	erase-eof
KEY_ENTER	enter
KEY_HOME	cursor home
KEY_IC	enter insert mode
KEY_LEFT	cursor left
KEY_NPAGE	scroll forward
KEY_F(1)	PF1
KEY_F(2)	PF2
:	
KEY_F(12)	PF12
KEY_PPAGE	scroll backward
KEY_RIGHT	cursor right
KEY_UP	cursor up

The uni-XEDIT keyboard test and keybind maintenance utility, uni-KEY, can assist you in determining the ***c-name*** of specific keys on your keyboard.

Both the **KEY** and the **COMMAND** operands are valid with either the ***seq*** or the ***c-name*** operand. Both **KEY** and **COMMAND** have required operands as discussed in the following paragraphs.

keyname is the name of the uni-XEDIT keyboard function that you are assigning to a key sequence or a curses key name. ***keyname*** may be any of the following:

backspace	cursor backspace (nondestructive)
backtab	cursor backtab to previous field
btab	cursor backtab to previous field
c_left	scroll window left one column
c_pgdn	scroll window forward one line
c_pgup	scroll window backward one line
c_right	scroll window right one column
dc	delete character at cursor
delete	delete character at cursor
down	cursor down
end	erase EOF
endl	erase EOF
enter	execute command and/or prefix command(s)
home	cursor to start of command line
ic	enter insert mode
insert	enter insert mode
left	cursor left
newline	cursor to beginning of next line
nop	no operation
npage	scroll window down one screen
pf1 ... pf24	PF1 through PF24
pgdn	scroll window down one screen
pgup	scroll window up one screen
ppage	scroll window up one screen
reset	reset insert mode
right	cursor right
tab	cursor tab to next field
tabchar	insert tab character (0x09 is inserted)
up	cursor up

cmd may be any valid uni-XEDIT command, including its required or optional operands.

SET KEYBIND commands are normally included in a uni-XEDIT profile. If you choose to do this, be sure to enclose the command in single or double quotes. This is necessary because key sequences contain characters such as “\” which are considered logical operators by the profile processor. uni-KEY automatically provides the required quotes.

When a key on your keyboard is mapped to a curses key name, that key automatically performs the uni-XEDIT function shown in the ***c-name*** table earlier in this section. For instance, the “Delete” key on many keyboards is automatically mapped to KEY_DC and performs the function of deleting individual characters in uni-XEDIT. If the labelled keys on your keyboard do not function automatically, the failure may result from any of the following circumstances:

- the key sequence transmitted by this key is not included in the terminfo database for your terminal type. Use uni-KEY to determine what key sequence is being transmitted. Use “SET KEYBIND ***seq*** KEY ***keyname***” or uni-KEY to map this key to the desired uni-XEDIT keyboard function.
- the key sequence transmitted by this key is mapped to a different curses key name from the one expected by uni-XEDIT. Use uni-KEY to determine the curses key name to which this key is mapped. Use “SET KEYBIND ***c-name*** KEY ***keyname***” to map this key to the desired uni-XEDIT keyboard function.
- the terminal type you are using is not transmitting a keycode for this key. When no keycode is transmitted, the key cannot be processed by the application.

For trouble-shooting keyboard mapping problems, additional items to consider include

- several keys on a keyboard that transmit the same keycode sequence. In this case, mapping one of these keys to a keyname or a command maps all of them to the same keyname or command.
- the effect of a terminal emulator program on the behavior of your keyboard. The terminal emulator program represents another layer of software between the key you press and uni-XEDIT's response to it. The native characteristics of a specific terminal emulator program may affect the screen display as well.
- vendor-specific modifications to the terminfo database for certain terminal types. While this is not always easy to determine, one known example is the IBM RS/6000 definition for the VT100 terminal type. Comments in the source file indicate modifications to this definition for a specific purpose and suggest the use of the VT100-AM definition for other uses.

The chapter entitled "A curses Application Primer: Terminal and Keyboard Issues in the Unix World" in the ***TWG System Administrator's Reference*** provides additional details to assist you.

Examples:

```
SET KEYBIND \e[146q KEY endl
```

assigns the ERASE-EOF function to the key that transmits \e[146q

```
'SET KEYB \e[146q KEY endl'
```

same as the previous example but enclosed in quotes for use in an edit profile

```
SET KEYB key_enter KEY newline
SET KEYB key_action KEY enter
```

assigns the newline function to the curses key name “KEY_ENTER” and assigns the enter function to the curses key name “KEY_ACTION”. This combination is useful on the IBM RS/6000 HFT terminal keyboard and allows you to use the Return key for “newline” and the Ctrl/Act key for “enter”.

```
KEYB \ej COMMAND down 3
```

assigns the keyboard sequence ESC-j to the command “DOWN 3"

```
KEYB ^x COMMAND qq
```

assigns the keyboard sequence CTL-x to the “QQUIT” command

```
SET KEYBIND C ^b KEY NEWLINE
```

assigns the newline function to the keyboard sequence CTL-b in character mode only

LINEND

The LINEND option enables or disables logical line-end processing and defines the logical line-end character. The logical line-end character allows you to stack a sequence of editor commands on the command line or on a single line in a macro. The line-end character, which separates the commands, mimics the behavior of the Enter key when line-end processing is enabled. When line-end processing is disabled, the character is treated as a normal character in the string.

[SET] LINENdON [*char*]
OFF

Initial default: ON #

ON enables line-end character processing and, optionally, specification of the logical line-end character.

char defines the logical line-end character and may be any special character such as &, *, $, or [. ***char*** may **not** be an alphabetic or numeric character. If ***char*** is omitted, the default character (#) is used.

OFF disables line-end character processing.

Examples:

```
set linend on @
```

enables logical line-end processing and defines "@" as the logical line-end character. After this command is executed, typing `down 3@clocate /abc/@cinsert def` and pressing Enter is identical to typing each command separately and pressing Enter at the end of each command.

`linen off`

disables logical line-end processing. After this command is executed, typing

```
down 3@clocate /abc/@cinsert
```

def and pressing Enter produces an “Invalid operand” message because the text string “3@clocate/abc/@cinsert def” is not a valid operand of the DOWN command.

LRECL

The LRECL option controls the logical record length used when a file is written to disk. All files are stored in variable-length record format (V). The LRECL setting defines the maximum record length.

[SET] LRecl ***n***

Initial default: LRecl 4096

n specifies the logical record length. ***n*** may be a positive integer less than or equal to 4096, or it may be * to set the LRECL to 4096.

MASK The MASK option controls the mask that appears in newly inserted data lines.

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

Initial default: blanks

Define allows you to define a new mask by modifying a template that appears on the command line. The template is identical to a scale line, making it easier to create column-oriented masks. Press Enter to save the new mask.

Immed defines a new mask in a single command.

text is the mask value. If ***text*** is omitted, the mask value is blanks.

Modify allows you to change the current mask. It displays the current mask on the command line for modification. Press Enter to save the new mask.

MSGLINE

The MSGLINE option controls the presence and location of the message line on the screen display.

[SET]	**MSGLine**	**ON** **OFF**	***pos***

Initial default: MSGLINE ON 2

Use **ON** to specify the presence of a message line on the display. If you specify **OFF**, no message line is present. When no message line is present, you will not see diagnostic messages or output from the EMSG, MSG, or QUERY command.

pos designates the location on the screen where the message line should appear. ***pos*** may be any of the following:

M	middle of the screen
M+*n*	*n* lines below the middle of the screen
M-*n*	*n* lines above the middle of the screen
n	*n* lines below the top of the screen
+*n*	same as *n*
-*n*	*n* lines above the bottom of the screen

SET MSGLINE may affect the results of the following commands:

EMSG
MSG
QUERY

Examples:

```
SET MSGLINE ON M+5
```

messages appear five lines below the center of the screen

```
MSGL ON -7
```

messages appear seven lines up from the bottom of the screen

MSGMODE The MSGMODE option controls the display of messages. Messages include editor diagnostic messages as well as output from the EMSG, MSG, and QUERY commands.

```
[SET]  MSGMode   ON   [LONG]
                      [SHORT]
                 OFF
```

Initial default: MSGMODE ON LONG

ON specifies that messages be displayed on the message line. **OFF** suppresses the display of all messages.

LONG specifies that all messages be displayed. **SHORT** suppresses the display of informational and error messages, but permits the display of output from EMSG, MSG, and QUERY.

SET MSGMODE may affect the results of the following commands:

EMSG
MSG
QUERY

NONDISP

The NONDISP option assigns a display character that appears in place of file data that does not have a normal character representation.

[SET] NONDISP [*char*]

Initial default: "

char is any displayable character. If ***char*** is omitted, the display character is set to blank.

Examples:

```
SET NOND @
```

sets the non-display character to "@"

NULLS

The NULLS option specifies whether trailing blanks on each line are displayed as blanks (hexadecimal 20) or as nulls (hexadecimal 00).

```
[SET]  NULls  ON
              OFF
```

Initial default: NULLS OFF

This option maintains an internal status but has no effect on the editor.

NUMBER

The NUMBER option controls the display of line numbers in the prefix area.

```
[SET]  NUMber   ON
                OFF
```

Initial default: NUMBER OFF

When **ON** is specified, sequential line numbers appear in the prefix area. The default form of the line number includes leading zeros. Use SET PREFIX NULLS in conjunction with SET NUMBER ON to suppress the leading zeros.

When **OFF** is specified, the prefix display defaults to five equal signs (=====).

PENDING

The PENDING option controls the execution of prefix macros and the appearance of the screen display during macro execution.

[SET]	**PENDing**	**ON**	***string***
		BLOCK	***string***
		ERROR	***string***
		OFF	

Initial default: none

ON displays ***string*** in the prefix area of the line where the macro name was typed. An entry is added to the pending list.

BLOCK displays ***string*** in the prefix area. ***string*** is the block form of a prefix macro. An entry is added to the pending list.

ERROR specifies the action taken when an error occurs during macro processing. It displays ***string***, preceded by a question mark (?), in the prefix area of the line where the macro name was typed.

OFF removes a pending prefix from the prefix area and from the pending list.

The pending list consists of prefix commands and macros that have not been processed. The pending list is updated with each new entry of a prefix command or macro. All entries are associated with specific lines in the file. Whenever the pending list is updated, all entries that are ready for execution are processed. Entries are removed from the pending list when they are executed or overtyped.

Prefix macros often use the information contained in the pending list to control their execution and screen display. ***Chapter 6: Edit Macros*** includes a complete discussion of prefix macros.

PF

The PF option allows you to define PF keys.

```
[SET] PFn     [BEFORE]     [string]
              [AFTER]
              [ONLY]
              [IGNORE]
```

Initial defaults:

PF1 and PF13	**BEFORE HELP**
PF2 and PF14	**undefined**
PF3 and PF15	**BEFORE QUIT**
PF4 and PF16	**BEFORE TABKEY**
PF5 and PF17	**BEFORE SCHANGE 6 (& SCHANGE 18)**
PF6 and PF18	**ONLY ?**
PF7 and PF19	**BEFORE BACKWARD**
PF8 and PF20	**BEFORE FORWARD**
PF9 and PF21	**ONLY =**
PF10 and PF22	**BEFORE RGTLEFT**
PF11 and PF23	**BEFORE SPLTJOIN**
PF12 and PF24	**BEFORE CURSOR HOME**

n is the number of the PF key to be defined.

With no additional operands, SET PF removes the current definition of the specified PF key.

string may be any uni-XEDIT line command. When specified, ***string*** defines a function to be performed by the PF key in addition to any pending command line or prefix input. ***string*** may also be TABKEY.

BEFORE specifies that ***string*** is executed prior to pending line or prefix commands.

AFTER specifies that ***string*** is executed after pending line or prefix commands.

ONLY specifies that only ***string*** is executed. Pending command line input is discarded. Pending prefix commands are executed. **ONLY** is the default if ***string*** begins with '?' or '='.

IGNORE specifies that ***string*** is not executed when command line input is present.

SET PF may affect the results of any line command.

Examples:

`SET PF4` removes the definition assigned to PF4

`SET PF4 ONLY macro dbs`

assigns the "dbs" macro to PF4. Entries on the command line are discarded when you presss PF4.

`PF4 BEFORE macro dbs`

assigns the "dbs" macro to PF4. When you press PF4, the "dbs" macro is executed before pending command line or prefix input.

POINT

The POINT option defines a symbolic name for the current line.

[SET] Point *.symbol* [OFF]

Initial default: none

.symbol is the symbolic name assigned to the current line. The name must begin with a period (.) and may be one to eight alphanumeric or special characters. You may assign more than one name to a single line by typing multiple SET POINT commands.

OFF removes the specified symbolic name from the current line.

You may also define symbolic names using prefix commands. ***Chapter 3: Prefix Commands*** provides further information.

Symbolic names may be used as targets in many line commands.

Examples:

```
SET POINT .whereami
```

assigns the symbolic name “.whereami” to the current line

```
P .whereami OFF
```

deletes the name “.whereami” from the current line

PREFIX

The PREFIX option allows you to control the presence, location, and appearance of the prefix area. It also allows you to define synonyms for prefix commands.

```
[SET]  PREfix  ON       [Left]
               Nulls    [Right]
               OFF

               Synonym  newname  oldname
```

Initial default: PREFIX ON LEFT

When you specify **ON**, the prefix area appears with each line of data. The default display is five equal signs (=====). Use SET NUMBER ON to display line numbers in the prefix area.

When you specify **NULLS**, the default prefix display is changed to five null characters. The presence of null characters in the prefix area allows you to enter prefix commands when you are in insert mode. Normally, entry of prefix commands is inhibited when you are in insert mode. Use SET PREFIX NULLS in conjunction with SET NUMBER ON to display line numbers without leading zeros in the prefix area.

OFF suppresses the display of the prefix area. All entry of prefix commands is inhibited.

Left and **Right** are valid only with **ON** and **NULLS**. **Left** positions the prefix area to the left of the data. **Right** positions the prefix area to the right of the data.

Synonym allows you to define synonyms for prefix commands or macros.

newname is the synonym assigned to an existing prefix. ***newname*** may be up to five alphabetic characters.

oldname is the prefix command or the name of the prefix macro for which you are defining a synonym. ***oldname*** may be up to eight characters long.

Examples:

`SET PREFIX ON RIGHT`	displays prefix area to the right of the data

`PREFIX ON L`
`NUMBER ON`
	displays prefix area to the left of the data, showing five digit line numbers with leading zeros

`PRE NULLS`	displays null characters in the prefix area
`PRE OFF`	no prefix area displays
`PRE SYNONYM a f`	defines “a” as a synonym for the “f” prefix command
`PRE S u upper`	defines “u” as a synonym for the prefix macro named “upper”

RANGE

The RANGE option defines the range of lines to be affected by edit operations. This creates a subset of the file for both display and command purposes.

[SET] RANGe *target-1 target-2*

Initial default: Top of File   Bottom of File

target-1 defines the top of the range. ***target-2*** defines the bottom of the range. Both ***target-1*** and ***target-2*** may be

- an explicit number
- a displacement
- a symbolic name
- a string

When you use SET RANGE to create a subset of the file, the top and bottom lines are changed to “Top of Range” and “Bottom of Range”, respectively.

SET RANGE may affect the results of any line command or prefix command.

Examples:

```
SET RANGE 1 50
```

line 1 is the top of the range and line 50 is the bottom of the range. Lines 51 through the end of file are not affected by line or prefix commands and do not appear in the display.

```
RANGE 10 *
```

line 10 is the top of the range and the bottom of the file is the bottom of the range

```
RAN -* *
```

resets the range to the physical top of file and bottom of file

RECFM

The RECFM option defines the record format of a file.

[SET]	**RECFm**	**V**
		F
		FP
		VP

Initial default: RECFM V

This option maintains an internal status but has no effect on the editor. All Unix files are in variable format.

RESERVED

The RESERVED option reserves lines on the uni-XEDIT display screen which cannot be used by the editor. In addition, it defines the display characteristics of reserved lines and specifies the text that is to appear on them. It is intended for use in a macro.

[SET]	**RESERved**	***pos***	**High**	**[*text*]**
			Nohigh	
			Off	

Initial default: Off

pos designates the location on the screen where the special line is to be reserved. ***pos*** may be any of the following:

M	middle of the screen
M+*n*	*n* lines below the middle of the screen
M-*n*	*n* lines above the middle of the screen
n	*n* lines below the top of the screen
+*n*	same as *n*
-*n*	*n* lines above the bottom of the screen

High specifies that data on the reserved line is highlighted. Although highlighting is not supported in this release of uni-XEDIT, the syntax is supported for macro compatibility.

Nohigh specifies that data on the reserved line is not highlighted.

Off returns a previously reserved line for use by the editor.

text is the data to be displayed on the reserved line.

Although color, extended highlighting, and use of alternate program symbol sets are not supported in this release of uni-XEDIT, the syntax is supported for macro compatibility.

Examples:

These examples assume that the following SET CTLCHAR commands have been executed prior to the SET RESERVED:

```
    set ctlchar % escape
    set ctlchar P protect
    set ctlchar @ noprotect
    set ctlchar I noprotect invisible
```

The discussion of CTLCHAR in this chapter describes the purpose and effect of these commands.

```
set reserved 5 nohigh %PName:  %@             %P
```

Reserve line 5 for display of a field containing the text string “Name: ” followed by a field containing blanks. The first field is protected from user input. Data entry is permitted in the second, unprotected field.

```
reser 6 n %PEnter password:  %I       %P
```

Reserve line 6 for display of a field containing the text string “Enter password: ” followed by a field containing blanks. The first field is protected from user input. Data entry is permitted in the second, unprotected field. Because the display charactistics of the second field are defined as “Invisible”, the data that is typed in this field does not appear on the screen.

The section entitled “User Applications Based on uni-XEDIT” in ***Chapter 6: Edit Macros*** contains an example of the use of SET RESERVED. In addition, the uni-XEDIT Sample Library, distributed with editor, includes an example of the use of reserved lines in a uni-XEDIT based application. The adbook application uses two edit macros in which SET RESERVED is used to configure the screen display.

SELECT

The SELECT option assigns a selection level to lines in a file. It is used in conjunction with the DISPLAY, SCOPE, and SHADOW options to control which lines appear on the screen display.

```
[SET] SELect  [+]  n  [target]
              [-]
```

Initial default: all lines set to selection level 0

n is a positive number that indicates the selection level to be assigned. Only one selection level may be assigned to a line. When the optional sign is omitted, ***n*** is an absolute selection level.

+ and **-** are used to modify the current selection level. When **+** is specified, ***n*** is added to the current selection level to assign the new selection level. When **-** is specified, ***n*** is subtracted from the current selection level.

target defines range of lines to which this selection assignment applies. The specified selection level is assigned to all lines beginning with the current line up to, but not including, the line identified by ***target***. If ***target*** is specified as "*", the range begins with the current line and extends to the end of the file. If ***target*** is omitted, the specified level is applied to the current line.

SELMODE

The SELMODE option controls the format in which text is selected. In graphical mode, SELMODE affects text selection using a pointing device (mouse) or equivalent keyboard navigation sequences. In character mode, SELMODE affects the results of the MARK command.

```
[SET] SELMODE   BLOCK
                COLUMN
                LINE
                STREAM
```

Initial default: STREAM

BLOCK defines the text selection format as an arbitrary rectangular block. The block is identified by marking its diagonally opposite corners.

COLUMN defines the text selection format as a rectangular block from top of file to bottom of file spanning specific columns. The block is identified by marking the beginning and ending columns. The cursor may be positioned on any line of data to mark the columns.

LINE defines the text selection format as a rectangular block of complete lines. The block is identified by marking the beginning and ending line. The cursor may be positioned anywhere within the line to be marked.

STREAM defines the text selection as an arbitrary block. It may begin at any point on a line and end at any point on the same line or a different line. It includes all complete lines between the beginning point and the ending point. The block is identified by marking the beginning point and the ending point.

SCALE

The SCALE option controls the presence and location of the scale line on the screen display.

[SET]	**SCALe**	**ON**	***pos***
		OFF	

Initial default: SCALE ON M+1

ON displays the scale line on the screen. **OFF** removes the scale line from the screen.

pos designates the location on the screen where the scale line should appear. ***pos*** may be any of the following:

M	middle of the screen
M+*n*	*n* lines below the middle of the screen
M-*n*	*n* lines above the middle of the screen
n	*n* lines below the top of the screen
+*n*	same as *n*
-*n*	*n* lines above the bottom of the screen

Examples:

```
SET SCALE ON m-5
```

places the scale line 5 lines above the middle of the screen

`SCAL OFF` removes the scale line from the display

SCOPE

The SCOPE option defines the set of lines in the file to be affected by editor operations.

[SET]	**SCOPE**	**Display**
		All

Initial default: SCOPE DISPLAY

When **Display** is specified, editor operations affect only lines that are currently displayed. When **All** is specified, editor operations affect both displayed lines and those not currently displayed.

The SCOPE setting does not affect lines that are outside the current RANGE.

SET SCOPE may affect the results of any line command or prefix command.

SCREEN

The SCREEN option allows you to partition your screen display into multiple windows. This feature lets you view multiple parts of a single file or multiple files in the edit ring simultaneously.

[SET] SCReen *n* [Horizontal]
[Vertical]

Initial default: SCREEN 1 HORIZONTAL

n specifies the number of screens to display. Without additional operands, SET SCReen ***n*** partitions the display into ***n*** horizontal screens. When ***n*** is specified as “1", the display reverts to a single screen.

Horizontal specifies that the screen be split horizontally. **Vertical** specifies that the screen be split vertically.

When you are editing a single file, that file appears in all split screens. The results of line commands or prefix commands appear in all screens. The individual prefixes of block prefix commands may be typed in different screens. Similarly, the destination prefix for a copy or move prefix may be typed in a different screen from the C or M prefix. Only one screen may be in input mode at any time. All other windows on the file remain in edit mode. Scrolling is independent in each window.

If there are multiple files in the edit ring, they are distributed among the split screens in the order they appear in the ring. When a different file appears in each screen, line commands and prefix commands affect only the screen in which they are entered. Multiple screens may be in input mode simultaneously.

The maximum value for ***n*** is determined by your original screen size. uni-XEDIT requires that there be at least five lines in each horizontal window or at least twenty characters in each vertical window.

SHADOW

The SHADOW option controls the appearance of shadow lines to represent lines that have been excluded from the display.

```
[SET]  SHADow     ON
                  OFF
```

Initial default: SHADOW ON

ON displays a shadow line indicating the number of lines excluded from the display at this location. Shadow lines appear as

```
---------n line(s) excluded---------
```

where ***n*** is the number of lines excluded.

OFF suppresses the display of shadow lines to represent excluded lines.

SET SHADOW affects the display resulting from the ALL line command and the X (Exclude) prefix command.

SHELLTERM The SHELLTERM option controls the window that appears in graphical mode when the CMS, SHELL, or ! command is used without operands. It has no effect in character mode.

[SET] SHELLTERM ***type* [*opts*] -e twgpause**
NOTERM

Initial default: xterm -e twgpause

type is any valid terminal type provided by your window manager. ***type*** must support the "-e" option to execute a program.

opts may be any valid terminal options for the ***type*** specified. The documentation for your window manager provides details on valid terminal options for the terminal types it supports.

-e twgpause is required and must be specified in lowercase. This executes the twgpause utility which permits the terminal window to remain displayed until you respond to the message "Press ENTER to return to the editor".

Use **NOTERM** to suppress the display of a terminal window if you do not wish to see the output from shell commands. If **NOTERM** is specified, temporary access to the shell for a series of commands is disabled.

SPAN

The SPAN option specifies whether characters in string targets may span multiple lines.

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

Initial default: SPAN OFF BLANK 2

When **ON** is specified, string targets may span multiple lines. Lines in the file are concatenated to determine if a match for the string is found.

Specify **Blank** to insert a blank character between concatenated lines. Specify **Noblank** to concatenate lines without separation. **ON** with no additional operands is equivalent to ON BLANK 2.

n specifies the number of consecutive lines that may be concatenated to search for the string target. ***n*** may be a number, or it may be * to include all lines in the file.

When **OFF** is specified, characters in string targets must all occur on a single line.

SET SPAN may affect the results of any command that supports a string target.

Examples:

Data lines:
```
This text will
be found
```

Command:
```
L /will be/
```

SPAN setting:	ON BLANK 2
Results:	current line is not moved. “will be” is not found within two consecutive lines.

SPAN setting:	ON BLANK 3
Results:	current line is moved because “will be” is found within three consecutive lines

SPAN setting:	ON NOBLANK
Results:	current line is not moved because there is no space between “will” and “be” in the concatenated lines

SPILL

The SPILL option controls whether data that would extend beyond the truncation column is spilled onto one or more new lines in the file or is lost.

```
[SET]  SPILL  ON
              WORD
              OFF
```

Initial default: SPILL OFF

ON specifies that data that would extend beyond the truncation column be inserted as one or more new lines of data in the file. The first character that would extend beyond the truncation column becomes the first character on the first inserted line.

WORD is similar to **ON** except that the data is split only where a blank space occurs. All characters beyond the last blank space before the truncation column are inserted on the next line.

OFF specifies that data beyond the truncation column be truncated (lost).

SET SPILL may affect the results of the following commands:

CAPPEND
CHANGE
CINSERT
COVERLAY
CREPLACE
GET
INPUT
REPLACE
SHIFT

STAY

The STAY option controls the movement of the current line with commands that support string targets.

[SET] STAY ON
OFF

Initial default: STAY OFF

ON indicates that the current line remain in its original location. **OFF** indicates that the current line is repositioned.

SET STAY affects the position of the current line when the target search is **not** successful for the following commands:

LOCATE
FIND
FINDUP

SET STAY affects the position of the current line when the target search **is** successful for the following commands:

CHANGE
COMPRESS
EXPAND
LOWERCAS
SHIFT
UPPERCAS

SYNCSCROLL The SYNCSCROLL option controls synchronized scrolling of multiple windows.

```
[SET] SYNCSCROLL     ON          [V]
                                 [H]
                     OFF
```

Initial default: OFF

ON enables synchronized scrolling.

V synchcronizes scrolling vertically. **H** synchronizes scrolling horizontally. If both **V** and **H** are omitted, scrolling is synchronized in both directions.

OFF disables synchronized scrolling.

SYNONYM

The SYNONYM option controls synonym processing and has two forms. The first form turns synonym recognition on or off.

[SET] SYNonym ON
OFF

Initial default: SYNONYM ON

ON indicates that uni-XEDIT should search for synonymns. **OFF** indicates that uni-XEDIT should ignore synonyms.

The second form is used to define synonyms for line commands or macros.

[SET] SYNonym *newname* [*n*] *oldname*

newname is the synonym being defined. ***newname*** may be one to eight alphabetic characters.

oldname is the name of the existing line command or macro.

n defines the minimum number of characters to be recognized as an abbreviation for this synonym. When ***n*** is omitted, you must type the full synonym name.

Examples:

```
SET SYNONYM e xedit
```

defines "e" as a synonym for the XEDIT command

```
SYN finis 3 bottom
```

defines "finis" as a synonym for the BOTTOM command. "finis" may be abbreviated as "fin".

TABLINE

The TABLINE option controls the presence and location of a tab line in the screen display. A tab line identifies the current tab stop settings with a “T” at each tab stop.

[SET]	**TABLine**	**ON**	***pos***
		OFF	

Initial default: TABLINE OFF -3

ON displays the tabline on the screen. **OFF** removes the tabline from the screen.

pos designates the location on the screen where the tab line should appear. ***pos*** may be any of the following:

M	middle of the screen
M+*n*	*n* lines below the middle of the screen
M-*n*	*n* lines above the middle of the screen
n	*n* lines below the top of the screen
+*n*	same as *n*
-*n*	*n* lines above the bottom of the screen

The tab line may also be displayed using the TABL prefix command, which is discussed in ***Chapter 3: Prefix Commands.***

TABS

The TABS option controls tab processing and has two forms. The first form enables or disables the action of TAB key.

```
[SET]  TABS   ON
              OFF
```

Initial default: TABS OFF

When **ON** is specified, the TAB key moves the cursor to the next tab stop. When **OFF** is specified, the TAB key moves the cursor to the beginning of the next input field.

The TABS setting has no effect on the action of the PF key defined as TABKEY.

The second form allows you to define tab stops for a file.

```
[SET]  TABS   tab1  [tab2  [. . . tabn]]
              INCR tab#
```

Initial default: 1 5 10 15 20 ... 110

tab is the column number for each tab stop specified. With this syntax, you must specify an explicit number for each tab stop.

The **INCR** operand uses ***tab#*** to define the number of character positions between each tab stop. This syntax establishes tab stops across the entire width of the file.

Examples:

```
SET TABS 10 30 40
```

sets tab stops at columns 10, 30 and 40 only

```
TABS INCR 8
```

sets tab stops every 8 positions, beginning at position 0

TRUNC

The TRUNC option defines the truncation column, which is the last column in a line on which data may be entered or modified.

[SET] TRunc [*n*]

Initial default: TRUNC 4096

n specifies the column number at which data is truncated or spilled. ***n*** may be a number, or it may be * to reset the truncation column to 4096.

Both the truncation column and the *zone-2* column limit the data searched to match string operands. For this reason, the *zone-2* column cannot be larger than the truncation column. Using SET TRUNC to make the truncation column smaller than the *zone-2* column automatically resets the *zone-2* column. Using SET TRUNC to make the truncation column larger than the *zone-2* column does not reset the *zone-2* column.

VARBLANK

The VARBLANK option controls whether or not the number of blanks between words is significant for target searches.

[SET] VARblank ON
OFF

Initial default: VARBLANK OFF

When **ON** is specified, multiple blanks between words are treated as a single blank and are therefore not significant for a target search.

OFF indicates that the number of blanks between words is significant during the search.

SET VARBLANK may affect the results of any command that supports a target operand.

Examples:

Data line:	`Three   blanks   between words`
Command:	`L /Three blanks/`
VAR setting:	ON
Results:	current line is repositioned
VAR setting:	OFF
Results:	current line is not repositioned

VERIFY

The VERIFY option controls the columns that appear on the screen. It further controls whether the display is ASCII or hexadecimal.

[SET] Verify [ON] [[Hex] *disp-field1* [... *disp-fieldn*]]
[OFF]

Initial default: VERIFY OFF 1 4096

The **ON** and **OFF** operands normally control the display of changed lines on the message line. This feature is not yet available in uni-XEDIT but will be implemented in a future release. The syntax is processed without generating an error, but no display of changed lines occurs.

Hex changes the display to hexadecimal representation rather than ASCII characters.

You may specify one or more display-fields to appear on the screen. A ***disp-field*** consists of a pair of column numbers that specify the beginning and end of the field, respectively. When used as the ending column number, * is synonymous with the maximum width of the file.

Example:

```
SET VERIFY 1 15 35 50
```

restricts the display to columns 1-15 and columns 35-50. Data in columns 16-34 and beyond column 50 does not appear on the screen.

```
V H 1 25
```

displays only columns 1-25 in hexadecimal representation

WRAP

The WRAP option controls the action taken when a string search encounters the beginning or end of the file or the range.

[SET]	**WRap**	**ON**
		OFF

Initial default: WRAP OFF

When **ON** is specified, the search automatically “wraps” through the data until the string is found or the starting point is reached.

When **OFF** is specified, the search terminates at the beginning or end of the file or the range if the string is not found.

SET WRAP may affect the results of the following commands:

CLOCATE
FIND
FINDUP
LOCATE

ZONE

The ZONE option limits the columns scanned for target searches.

[SET] Zone *zone-1* *zone-2*

Initial default: ZONE 1 4096

zone-1 is the number of the leftmost column to be scanned. ***zone-2*** is the number of the rightmost column to be scanned. Specifying * as ***zone-2*** sets the right zone at the truncation column.

Both the ***zone-2*** column and the truncation column limit the data searched to match string operands. For this reason, the ***zone-2*** column cannot be larger than the truncation column. Using SET TRUNC to make the truncation column smaller than the ***zone-2*** column automatically resets the ***zone-2*** column. Using SET TRUNC to make the truncation column larger than the ***zone-2*** column does not reset the ***zone-2*** column.

Chapter 6: Edit Macros

An edit macro is a program that extends the capabilities of the editor. Edit macros are used to

- customize the editor display or environment (usually an editor profile)
- extend the editor’s line and prefix command sets
- automate standard processes

This chapter describes the creation and use of edit macros. Early sections discuss the general requirements, including contents, naming conventions, and execution considerations. Subsequent sections describe the use of macros for specific purposes and present examples of such macros.

Macro Contents

An edit macro is a sequence of statements to be executed to accomplish a specific task. These statements may be

- uni-XEDIT line commands
- Unix system commands
- REXX program statements
- any combination of these elements

A macro that contains only uni-XEDIT line commands may be executed from any uni-XEDIT session. If a macro contains any of the other types of statements, its execution requires a license for uni-XEDIT Extended.

uni-XEDIT Extended includes an embedded REXX interpreter to process these statements.

Examples:

```
'set cmdline top'
'set msgline on 3'
'set scale off'
'set pf2 only qquit'
```

a macro that contains only uni-XEDIT line commands. This macro customizes the screen display and assigns the QQUIT command to PF2. Note the use of quotes to insure that the macro interpreter does not attempt variable substitution or arithmetic operations.

```
address unix 'make -f mkmyprog'
address unix 'myprog'
```

a macro that contains only Unix system commands. This macro creates a current version of a program named "myprog" and then executes it. Note the use of "address unix" to identify which external environment should receive these commands for processing. Without "address unix", these commands would be sent to the default external environment, uni-XEDIT.

```
nextupd = linein('next.update')
today = date('c')
if today < nextupd then do
   say 'No updates required'
   say 'Next updates scheduled on:' nextupd
   exit
   end
```

a macro that contains only REXX statements. This macro retrieves the date of the next scheduled update from the file "next.update", compares today's date to the date of the next scheduled update, and advises the user if no updates are required.

```
'MSGMODE OFF'                  /* suppress messages */
TOP'               /* start at the top of the file */
DO UNTIL EOF.1 = 'ON'
/* If you are not at "Bottom of File", do ..   .   */

   'LOCATE /Completed:/'
/*   Reposition the current line on the next       */
/*   line that contains "Completed:"               */

  'EXTRACT /CURLINE'   /* Get current line settings */
  'IF WORDS(CURLINE.3) > 1 THEN DO
/*   If the data on the current line contains      */
/*   more than one word (token), do...             */

      'L -/Order Number:/'
/*      Reposition current line on the prev-       */
/*      vious line that contains "Order Number:"   */

      'PUTD /***/ shipped.today'
/*      Write to a file named "shipped.today" all  */
/*      lines from the current line up to, but */
/*      but not including, the first line that     */
/*      contains "***".  Delete those lines from   */
/*      the file                                   */

     'PUTD 1 shipped.today'
*      Write and delete one more line             */

    END                    /* end of inner loop */
  'DOWN 1'          /* Move current line down one */
  'EXTRACT /EOF'           /* Extract EOF setting */
  END                        /* end of outer loop */
'MSGMODE ON'
ADDRESS UNIX 'lpr shipped.today'     /* print file */
ADDRESS UNIX 'rm shipped.today'     /* delete file */
```

a macro that combines all elements. This macro is used when editing a file of orders pending to migrate completed orders to a file of today's shipments, print that file, and then delete it. Comments are contained within pairs of /* */. Executable statements in the macro are shown in uppercase to distinguish them from the comments.

Macro Names

A macro is stored in an ordinary disk file, which may have any valid Unix file name. You may use local naming conventions to identify the file as a uni-XEDIT macro. You may also use naming conventions to easily recognize groups of related macros.

Examples:

```
printit
printit.xedit
print.shiplist
print.ship.list
```

several possible names for the shipping list macro shown in the previous example

```
orders.new
orders.update
orders.print.shiplist
orders.invoice
```

names for a group of macros that perform different functions for the order processing system

```
c.xedit
cobol.xedit
fortran.xedit
data.xedit
```

names for a group of profile macros that establish specific editor configurations based on the type of file to be edited

When you name macros using either of the following forms

name
***name*.xedit**

you can execute them simply by typing name.

uni-XEDIT enforces only one naming convention. The special name

```
.profile.xedit
```

is reserved for the default edit profile that is automatically executed by the “xe” editor invocation command. This special macro **must** reside in the user’s HOME directory. If a file named “.profile.xedit” is not found in the user’s HOME directory, the editor looks for a system-level profile named “.profile.xedit” in the same directory where the uni-XEDIT binary is installed.

Macro Libraries

Both naming conventions and the Unix file system structure provide options for organizing the storage of your macros. The previous section entitled “Macro Names” provides examples of using naming conventions for this purpose.

The Unix file system structure is particularly well suited for organizing libraries of macros. By setting up a directory for each library, you can organize your macros according to their use within the organization.

A set of macros used by everyone might be stored together in a directory such as /usr/local/xedit/macros. Macros for use with an order processing system might be stored in the directory where the order processing data is maintained or in a subdirectory of that directory. Macros for your own personal use might be stored in your HOME directory.

Regardless of your choice for organizing macro libraries, uni-XEDIT must know where to search for a macro to be executed. This is accomplished through the use of a special Unix environment variable, XEDITPATH. The section entitled “Macro Execution” contains a complete discussion of how the editor locates a macro for execution, including instructions for setting the XEDITPATH environment variable.

Macro Execution

A macro may be executed in any of the following ways:

- as an edit profile
- as a line command
- as a prefix command
- as the operand of the MACRO line command
- within another macro

The **-Prof** operand of the editor invocation command (xe) allows you to specify the name of a macro to be executed as an edit profile. From the Unix prompt, the command

```
    xe orders.data -p orders.new
```

initiates an edit session with the file “orders.data”, using the macro “orders.new” as a profile. Similarly, you may use the XEDIT line command to add a file to the ring with its own profile by typing

```
    x orders.history -p history
```

To execute a macro as a line command or as a prefix command, simply type the name of the macro on the command line or in the prefix area, respectively.

It is possible to have a macro whose name is identical to a uni-XEDIT command or a synonym that has been set for the current edit session. In this case, use the MACRO line command to explicitly execute the macro. Typing

```
    macro printit
```

on the command line is identical to typing

```
    printit
```

provided there is no synonym named “printit” set for this session.

In general, you **must** use the MACRO command if the name of your macro contains periods. The single exception is for macros with names in the form

name.xedit

where ***name*** does not include any periods. You may execute a macro named “printit.xedit” by typing any of the following on the command line:

```
printit
macro printit
macro printit.xedit
```

If two macros, named “printit” and “printit.xedit” exist in the same directory, you **must** use the last form to execute “printit.xedit”. Both of the first forms will execute “printit”.

There is no equivalent to the MACRO command for prefix macros. You must therefore be careful in the naming of prefix macros and prefix synonyms to avoid confusion.

You may execute a macro from within another macro by including it as if it were an editor command. You may also use the MACRO command within a macro to eliminate possible confusion. Prefix macros, like prefix commands, are not valid in a macro.

To locate a macro for execution, uni-XEDIT uses the following search order:

- current active directory
- directories specified in the environment variable XEDITPATH, in the order that they appear in the path list

The environment variable XEDITPATH is similar to the environment variable PATH in that it specifies one or more directories to be automatically searched. Unlike PATH, it is used only by uni-XEDIT.

The command(s) used to define XEDITPATH are dependent on the Unix shell that you use. Choose the appropriate command(s) from the following table:

Shell	Command(s)
C	setenv XEDITPATH *path*
Bourne	XEDITPATH=*path* export XEDITPATH
Korn	XEDITPATH=*path* export XEDITPATH

In all the previous examples, ***path*** is a list of one or more directory names, separated by colons (:). Thus, to define under the C shell an XEDITPATH that includes /usr/local/xedit and /home/sales/orders, type

```
setenv XEDITPATH /usr/local/xedit:/home/sales/orders
```

Now when you execute the macro “printit”, uni-XEDIT searches your current active directory, /usr/local/xedit, and /home/sales/orders – in that order – to locate the disk file “printit” or “printit.xedit”. If both files exist somewhere in the search path, the first one encountered is executed. If both files exist in the same directory, the one named “printit” is executed. If neither file is found, the following message appears:

```
    542e, No such subcommand: printit
```

You may bypass the search order for macros by using the MACRO command and providing the full path name of the macro to be executed. If the “printit” macro is located in the directory “/home/sales/programs”, which is neither the current directory nor an element in your XEDITPATH, you can still execute it by typing

```
MACRO /home/sales/programs/printit
```

The **-Prof** operand of both the “xe” invocation command and the XEDIT line command also supports the use of full path names to designate the profile macro to be executed.

Customizing the Editor Environment

The numerous options of the uni-XEDIT SET command allow you to customize the screen display, keyboard mappings, PF key definitions, and a variety of additional components of the editing environment. These options are discussed in detail in ***Chapter 5: SET Command Options***.

Once you have decided how to customize your editing environment, you will likely want to automate this process. Creating a macro that includes all the commands required to set up a custom configuration allows you to create this environment quickly at any time.

You may invoke a macro to modify or customize the editing environment at any time by typing its name on the command line alone or as the operand of the MACRO command. Often, however, such macros are used as editor profiles to customize the environment as part of initializing the edit session.

Profile macros may be executed in one of two ways:

- automatically
- explicitly, using the **-Prof** operand of the “xe” editor invocation command

For automatic execution, a profile macro must meet the following criteria:

- the name must be “.profile.xedit”
- the file must reside in the user’s HOME directory

For explicit execution using the **-Prof** operand of “xe”, a profile macro may reside anywhere in the Unix file system. This includes directories that are mounted as NFS links to file systems on other workstations in the network. If the location of your profile macro either is the current directory or is included in the definition of the environment variable XEDITPATH, you need specify only the name of the macro to use as a profile. If the location is not defined in XEDITPATH, you must provide the full path to the file. The section “Macro Libraries” in this chapter contains details on defining and using the XEDITPATH environment variable.

Examples:

```
xe data.file
```

initiates the edit session for a file named "data.file". If a file named ".profile.xedit" exists in this user's HOME directory or in the directory where the uni-XEDIT binary is installed, it is automatically executed to configure the edit environment. If such a file does not exist, the uni-XEDIT default settings are used.

```
xe data.file -p orders
```

initiates the edit session for a file named "data.file" and invokes the profile named "orders.xedit" to customize the environment. The file "orders.xedit" must be in the current active directory or in the XEDITPATH.

```
xe data.file -p /home/sales/macros/orders
```

initiates the edit session for a file named "data.file" and invokes the profile named "orders.xedit" to customize the environment. The specific profile named is executed regardless of the current active directory or the setting of XEDITPATH.

```
x shipments.file -p ship
```

uses the line command XEDIT to add a file named "shipments.file" to the edit ring. This file requires a different profile, "ship.xedit", from the one(s) in use for other files in the ring. The file "ship.xedit" is located either in the current active directory or in a directory specified by XEDITPATH.

Example Profile Macros:

```
'set cmdline top'
'set msgline on 3'
'set curline on 4'
'set scale off'
'set number on'
```

This profile changes the positions of the Command line, the Message Line, and the Current Line. It also removes the Scale Line from the display and displays line numbers in the prefix area.

```
'set keybind key_enter key newline'
'set keybind key_action key enter'
'set keybind \e[Z key backtab'
'set keybind \e[146q key endl'
'set PF2 only macro dbs'
```

This profile is suitable for the HFT keyboard on the IBM RS/6000. It re-maps the Enter key for use as "newline" and the Ctrl/Act key for use as "enter". It also sets up the proper mapping for the backtab and end keys and defines PF2 to execute the "dbs" (destructive backspace) macro found in the uni-XEDIT Sample Library.

```
'set keybind \e[208z key pf10'   /* R1 key  */
'set keybind \e[209z key pf11'   /* R2 key  */
'set keybind \e[210z key pf12'   /* R3 key  */
'set keybind \e[211z key insert' /* R4 key  */
'set keybind \e[213z key reset'  /* R6 key  */
'set keybind \e[214z key home'   /* R7 key  */
'set keybind \e[216z key pgup'   /* R9 key  */
'set keybind \e[220z key endl'   /* R13 key */
'set keybind \e[222z key pgdn'   /* R15 key */
```

This profile is suitable for all Sun Microsystems keyboards. It maps the R-keys to appropriate uni-XEDIT functions. For OpenWindows, this profile assumes the presence of the file ".ttyswrc" at the time the window was opened.

Command Set Extensions

Edit macros allow you to further customize the editor by extending the command set. You may create both line commands and prefix commands that perform functions not represented in the standard command set.

Line Command Set Extensions

A macro that is used to extend the line command set is any macro that is invoked from the command line. Such macros may also be assigned to a PF key using SET PF or mapped to other keys using SET KEYBIND. Like some standard line commands (for example, CURSOR HOME), some macros may be best used when associated with a particular key.

The uni-XEDIT Sample Library, which is included on the product distribution media, contains a number of edit macros that extend the line command set. These include a spell-check macro, a macro to perform destructive backspace, and a macro to move the cursor to the end of the data on the current line. The last two are best used when assigned to a specific key. Each of the macros in the library is fully commented. If a macro contains REXX programming statements, you must have a license for uni-XEDIT Extended to execute it.

Prefix Command Set Extensions

A macro that is used to extend the Prefix Command set is any macro that is invoked from the prefix area. Like standard prefix commands, prefix macros

- have very short names
- may accept an optional numeric operand
- may operate on a single line or a block of lines

Prefix command processing differs from line command processing in the following ways:

- optional operands immediately follow the command with no delimiting space
- block operations require the presence of a pair of prefixes before they can be executed. Unmatched block prefixes must therefore remain pending for later execution.
- operations on shadow lines are processed differently from operations on data lines
- a pending prefix may be removed without being executed

To accommodate these differences, special guidelines apply to the creation of prefix macros.

Prefix Macro Names. Prefix macros should be named using one of the following forms:

> ***name***
>
> ***name***.xedit

where *name* is one to eight alphanumeric characters, excluding periods. You may assign short names to prefix macros so that the command typed in the prefix area is identical to the name of the macro. Alternatively, you may assign longer, more descriptive names to prefix macros and use SET PREFIX SYNONYM to define a short name for typing in the prefix area. (***Chapter 5: SET Command Options*** contains the complete syntax for SET PREFIX.)

Some caution is required in assigning names or syno-nyms to prefix macros. If the macro is designed to support an optional numeric operand, do not assign it a name that ends in numbers. As an example, a macro to uppercase one or more lines of data ("u.xedit") must allow you to type "u4" or "4u" to indicate that four lines are to be processed. If you assign a name or synonym to this macro that resembles its syntax, the macro will not operate as expected.

Argument String Processing. When you execute a prefix macro, uni-XEDIT automatically generates an argument string that is passed to the macro. This argument string is

```
PREFIX   SET        PLINE  OP1  OP2  OP3
         SHADOW
         CLEAR
```

Therefore, the first line in a prefix macro should be a -REXX statement to process this argument string.

The components of the argument string are:

PREFIX	verification that this is a prefix macro call
SET	one of three possible values for the second token of the argument string; specifies that the prefix was entered on a data line
SHADOW	one of three possible values for the second token of the argument string; specifies that the prefix was entered on a shadow line
CLEAR	one of three possible values for the second token of the argument string; specifies that a new prefix or blanks replaces the pending prefix command
PLINE	contains the line number in the file on which the prefix was entered
OP*n*	OP1, OP2, and OP3 are the optional operands of the prefix macro

Macro Name Processing. To support both single line (c) and block (cc) prefix command syntax within a single macro, it is necessary to be able to determine the name that was typed in the prefix area. This is accomplished using the REXX statement “parse source”. The sixth token returned by parse source is the name by which this macro was invoked. Thus the statement

```
    parse source . . . . . name .
```

sets the variable “name” to the name typed in the prefix area.

When a single macro supports different functions (such as the single line or block mode alternatives of a prefix macro), you must define a synonym for one of the command forms. Use SET PREFIX SYNONYM, discussed in detail in ***Chapter 5: SET Command Options***, to define such synonyms.

This technique can be used for any macro that performs more than one function and is called by different names. Use SET SYNONYM to define synonyms for macros that are not prefix macros.

The Pending List. Another special consideration in writing a prefix macro is the pending status of the prefix. The pending list contains all prefix commands and macros that have not been processed. Each entry in the list is associated with a specific line in the file. The pending list is updated with every new entry of a prefix command or macro. Whenever the pending list is updated, all entries that are eligible for execution are processed and removed from the pending list. Entries that are eligible for execution include single line prefixes, block prefixes when the required pair is present, and prefixes that require a target prefix when both prefixes are present. All other entries remain pending and remain in the list. A pending prefix may also be removed from the list if it is replaced with a different prefix, blanks, or standard prefix area characters (= or numbers). If a prefix macro produces a non-zero return code, pending list execution is halted and all remaining entries are pending until you press Enter.

SET PENDING allows you to add a prefix to or remove a prefix from the pending list. It also allows you to control the prefix area display when a prefix is pending. ***Chapter 5: SET Command Options*** contains a complete discussion of SET PENDING.

QUERY PENDING and EXTRACT PENDING allow you to examine the contents of the pending list and control macro processing based on the results. Both QUERY PENDING and EXTRACT PENDING have special syntax as described below.

QUERY PENDING displays the first entry in the pending list that matches the name specified or all entries in the pending list.

Query PENDing [BLOCK] [OLDNAME] *name*

BLOCK indicates that Query should search the pending list for block entries only.

OLDNAME indicates that the ***name*** specified is the original name of the prefix rather than the synonym used to execute it.

name is the name of the prefix command or macro to be located in the pending list. If ***name*** is specified as *, QUERY returns information for all entries in the pending list. Otherwise, QUERY returns information for the first entry that matches ***name***.

The output from QUERY PENDING has the following form:

```
Line n:' name`, Oldname=' `, OP1=' `, OP2=' `,
OP3=' `
```

n is the line number on which the pending prefix was entered. ***name*** is the prefix name as typed. If ***name*** is a synonym, the Oldname= field contains the original name of the prefix. The OP1=, OP2=, and OP3= fields contain the optional prefix operands, if any.

EXTRACT PENDING returns information about the first entry in the pending list or the first entry in the list that matches the specified name. The search may be limited to pending prefixes within a specific range of lines in the file.

<P9.5B>EXTRACT /PENDing [BLOCK] [OLDNAME] ***name*** **[*t1* [*t2*]]**

BLOCK indicates that information should be extracted for block prefixes only.

OLDNAME indicates that the ***name*** specified is the original name of the prefix rather than the synonym used to execute it.

name is the name of the prefix command or macro to be located in the pending list. If ***name*** is specified as *, EXTRACT returns information for the first entry in the pending list. Otherwise, EXTRACT returns information for the first entry that matches ***name***. When **OLDNAME** is specified, ***name*** must be the original name of the prefix command or macro rather than a synonym.

t1 and ***t2*** are targets that limit the search for pending prefixes to a specific range of lines in the file. The default value for ***t1*** is the first line of the file. The default value for ***t2*** is the last line of the file. If ***t1*** is specified as a string or a displacement, the location is calculated relative to the top of the file. If ***t2*** is specified as a string or a displacement, the location is calculated relative to ***t1***.

The EXTRACT command in ***Chapter 4: Line Commands*** contains a complete description of the variables returned by EXTRACT PENDING.

Example Prefix Macros:

The examples that follow illustrate three ways to write a prefix macro to uppercase a line of data. The examples progress from the simplest and least flexible to the most complex and most flexible. Comments in the examples provide additional documentation.

```
/* Macro u.xedit. Uppercase a line of data   */

arg . . pline . /*Process the argument string*/
/*                pline is the line where the*/
/*                prefix was entered         */
'extract /line '      /* Get line number of */
/*                              current line */
':'pline     /* Move the current line to the */
/*         line where the prefix was entered */
'uppercas'     /* Uppercase the current line */
':'line.1   /* Move the current line back to */
/*                      its original location */
exit 0
```

This simple macro provides a prefix alternative to the UPPERCAS line command. You then type “u” in the prefix area of a line to uppercase that line. It is not necessary to move the current line first as with the UPPERCAS line command.

```
/* Macro u.xedit.   Uppercase one or more lines
 *                  of data.
 *
 * SYNTAX:   u   uppercase one line
 *           un  uppercase “n” lines
 *           nu  uppercase “n” lines
 *
 */
arg . . pline op1 . /* Process argument      */
/*        pline is the line where the prefix */
/*        was entered; op1 is the optional   */
/*        numeric operand                    */
if op1 = ‘’ then op1 = 1    /* If no numeric */
/*        operand was supplied, then set “n” */
/*        to 1                               */
‘extract /line ‘      /* Get line number of */
/*                                current line */
‘:’pline     /* Move the current line to the */
/*         line where the prefix was entered */
‘uppercas’ op1        /* Uppercase “n” lines */
‘:’line.1   /* Move the current line back to */
/*                      its original location */
exit 0
```

This macro is an extension of the previous example that allows you to specify the number of lines to be uppercased.

The first two examples are very simple macros. Since they do not process blocks of data, there is no need to determine the name by which they were called or to be concerned with processing the pending list. Furthermore, they include no error checking.

The final example illustrates a more complete implementation of the “u” prefix macro that provides maximum flexibility for using the command and important error checking.

```
/* Macro u.xedit.  Uppercase a single line,
 *                 "n" lines, or a block of
 *                 lines.
 *
 * SYNTAX:   u   uppercase one line
 *           un  uppercase "n" lines
 *           nu  uppercase "n" lines
 *           uu  uppercase a block of lines
 *
 */
parse source . . . . . name .    /* Get name */
/*                  by which macro was called */
arg pref func pline op extra  /* Process the */
/*                           argument string */

/* Check to see where the prefix was entered
 * If not the prefix area, do error processing
 */

if pref \= 'PREFIX' then
   call error1 name 'valid in prefix area only'

/* Check the function to be performed.  If it's
 * CLEAR, we exit; if it's SHADOW, print a
 * message.  If it's not SET, error processing.
 */

if func = 'CLEAR' then exit
if func = 'SHADOW' then
   call error1 'Not valid on shadow lines'
if func \= 'SET' then
   call error1 'valid in prefix area only'

/* Check for extraneous junk                 */

if extra \= '' then
   call error 'Extraneous parameter:' extra

/* Choose processing based on length of name */

select
 when length(name) = 1 then do  /* if just u */
   if op = '' then op = 1     /*if no number,*/
/*                               set it to 1 */
   if datatype(op,'W') then    /* if integer */
     'command :'pline 'uppercas' op /* do it */
     else call error 'invalid operand:' op
   end

 when length(name) = 2 then do      /* if uu */
   if op \= '' then      /* no op supported */
     call error 'invalid operand:' op
   'extract /pending block' name' ':0 :'pline '

/* Is one of these already in the pending list?
 * If so, (pending.0 is non-zero) then do the
 * operation.  If not, put this one in the
 * pending list.
 */
   if pending.0 \= 0 then do
     'command :'pending.1 'set pending off'
     'command :'pending.1 'uppercas :'pline+1

/* Target for uppercas must be one line
 * greater than the last line marked with the
 * uu prefix since uppercas modifies lines up
 * to, but not including, the target line
 */
     end
    else do
     'command :'pline 'set pending block' name
     end

 otherwise call error 'Invalid macro synonym'
 end

/* Position cursor as desired                */

'cursor file' pline 'priority 30'
exit

/* Error handling routines                   */

error:
'command :'pline 'set pending error' name||op
error1:
parse arg message
'command emsg' message
exit
```

Note the extensive error checking in this macro. Note also the use of the uni-XEDIT COMMAND command to eliminate any possible confusion with macros or synonyms. This could be extended to all the line commands used in the macro. Note also the use of the "***target command***" technique to position the current line before executing a line command. To use the block form of this macro, you must define "uu" as a synonym for "u". Use

```
set pref s uu u
```

from the command line or in a profile macro.

User Applications Based on uni-XEDIT

Edit macros allow you to implement full-screen user appications based primarily on uni-XEDIT. With SET RESERVED you can reserve lines on the screen for special purposes and specify text to appear on these lines. With SET CTLCHAR you can define the the display characteristics of fields in reserved lines. With READ you can determine what the user typed on the screen before pressing Enter or a PF key. A macro that combines these features allows you to design screens for user input and to base additional processing on the input received.

You may further extend this concept to include a uni-REXX program that drives the processing by invoking the editor with the appropriate macros. In such cases, the user can type a single command (the name of the uni-REXX program) to invoke a complex application based primarily on uni-XEDIT capabilities.

The following pages contain a listing of a macro that illustrates the use of SET CTLCHAR, SET RESERVED, and READ to provide a full-screen user interface. A screen print of the display created by the macro is shown immediately after the listing. In the macro, the processing based on user entries has been omitted to facilitate comparing the display setup with the results.

```
/*
 * menu.xedit  -  macro to display file authorization menu
 */
'extract /lscreen'                                  /* get screen length */
if lscreen.1 < 20 then do                /* if too short, message & exit */
   say 'Menu requires a screen with at least 20 lines'
   say 'Your screen has' lscreen.1 'lines'
   say 'Terminating now - please reconfigure your screen and retry'
   exit
   end
'set msgline on' lscreen.1                     /* message line at bottom */
'set scale off'
'set cmdline off'
'set prefix off'
'cursor screen 8 2'                     /* set initial cursor position */
'set linend off'                           /* allow use of # character */
'set ctlchar @ escape'     /* escape character for control char sequence */
'set ctlchar % protect'       /* control character for protected fields */
'set ctlchar & noprotect'    /* control character for unprotected fields */
'set ctlchar * noprotect invisible'/*ctl char for unprot fld, hidden data*/
do forever
   call showmenu                        /* call screen display routine */
   'read nochange tag'                     /* get data entered by user */
   do queued()
      parse pull tag one two three  /* process each line entered by user */
      select
         when tag = 'RES' then do
           <processing based on what you got from reserved line >
         when tag = 'CMK' then < processing for keybound keys >
         when tag = 'CMD' then < processing for command line data>
         when tag = 'PFK' then if one = 3 then do        /* if PF3, quit */
            'command qquit'
            exit
            end
           else <processing for PF keys>
         when tag = 'ETK' then <processing for Enter key>
         otherwise nop
         end
      end
   <processing based on data entered by user>
   end
exit
showmenu:
'set reserved 1 nohigh'
'set reserved 2 nohigh'
'set reserved 3 nohigh @%             FILE AUTHORIZATION MENU           @%'
'set reserved 4 nohigh'
'set reserved 5 nohigh @% Type X beside each file set you wish to access@%'
'set reserved 6 nohigh'
'set reserved 7 nohigh'
'set reserved 8 nohigh @& @% Payroll               @%'
'set reserved 9 nohigh @& @% Accounts Receivable   @%'
'set reserved 10 nohigh @& @% Accounts Payable     @%'
'set reserved 11 nohigh @& @% General Ledger       @%'
'set reserved 12 nohigh @& @% Personnel            @%'
'set reserved 13 nohigh'
```

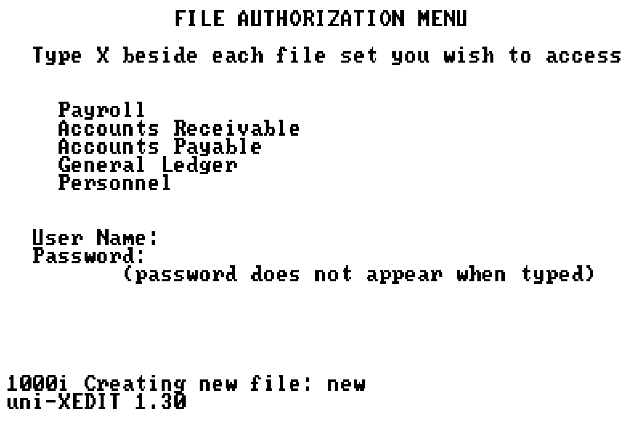
```
'set reserved 14 nohigh'
'set reserved 15 nohigh @% User Name:    @&               @%'
'set reserved 16 nohigh @% Password:     @*               @%'
'set reserved 17 nohigh @%       (password does not appear when typed) @%'
'set reserved 18 nohigh'
'set reserved 19 nohigh'
do j = 20 to lscreen.1-1
   'set reserved' j 'nohigh'
   end
return
```

To use this macro, invoke uni-XEDIT with the command

```
xe newfile -p menu
```

You may wish to place this invocation string in a shell script or a REXX program so that the user can type a single command at the system prompt.

The resulting screen display is shown below. The tab key moves the cursor automatically to the next input field.



```
              FILE AUTHORIZATION MENU
  Type X beside each file set you wish to access

    Payroll
    Accounts Receivable
    Accounts Payable
    General Ledger
    Personnel

  User Name:
  Password:
         (password does not appear when typed)



1000i Creating new file: new
uni-XEDIT 1.30
```

The uni-XEDIT Sample Library, included on the product distribution media, contains an electronic address book application that incorporates all these concepts. The main REXX program, adbook, invokes uni-XEDIT with a macro that permits full-screen data entry of new names and addresses. It also illustrates one technique for finding and displaying names stored in the address book.

Appendix A: Message Summary

The following is a list of messages that may be generated by uni-XEDIT.

000x	Internal error in routine *routine-name*
002e	File *filename* not found
003e	Invalid option: *option*
009e	Column *n* exceeds record length (*x*)
053e	Invalid sort field pair defined
054e	Incomplete fileid specified
062e	Invalid character in fileid *filename*
063e	No sort list given
104s	Error reading file *filename* from disk
105s	Error writing file *filename* to disk
156e	Record *n* not found—file *filename* has only *x* records
501i	*n* line(s) deleted
502i	*n* line(s) recovered
503e	spilled | truncated

504e	*n* spilled | truncated
505e	Not executed — the target line (*n*) is within the lines to move
506i	*n* line(s) moved|copied
507e	No preserved data to restore
510i	Autosaved as *filename*
511e	String2 contains more arbitrary characters than string1
515e	Recfm must be F, V, FP or VP
517i	*n* occurrences changed in *x* line(s)
519e	LRECL must be lower than WIDTH (*n*)
520e	Invalid operand: *operand*
521e	Invalid line number
525e	Invalid PFkey number
528e	Invalid range: target2 (line *n*) precedes target1 (line *x*)
529e	Subcommand is only valid in *mode* mode
530i	*n* file(s) in storage
533e	Line *n* is not reserved
534e	Too many logical screens defined
539e	Named line not found
540e	Name already defined on line *n*
541e	Invalid name

542e	No such subcommand: *command*
543e	Invalid number: *n*
545e	Missing operand(s)
546e	Target not found
547e	Synonym definition incomplete
548e	Invalid synonym operand *operand*
549e	Synonym abbreviation too large
550e	Too many operands in synonym definition
555e	File *filename* already in storage
557s	No more storage to insert lines
561e	Cursor is not on a valid data field
562e	No line(s) saved by PUT(D) subcommand
563w	Records spilled | truncated
564w	EOF reached
565w	EOF reached; records truncated
573i	Input mode:
575e	Invalid [argument or] zone column(s) defined
576e	Total offset exceeds LRECL (*n*)
577e	File has been changed; type QQUIT to quit anyway
585e	No line(s) changed
586e	Not found

588e Prefix command waiting...

592w Wrapped .....

594e File *filename* already exists; use FFILE/SSAVE

600e First selection level cannot be greater than second

659e Invalid prefix subcommand: *command*

661e Prefix *p* is invalid for the line on which it was entered

700e Logical AND operator & not valid for column targets

1000i Creating new file: *filename*

1001i Command mode:

1002i File is read only: *filename*

1003i Directory doesn't exist: *directory-name*

1004i File is write only: *filename*

1005i Days left until demo expires: *number*

Note: Message 1005i occurs only in demo or trial installations. *number* is the number of days remaining in the demo or trial period. The message begins appearing when 15 days are left in the period. The number is decremented each day for the remainder of the demo or trial. When a demo or trial period has expired, The Workstation Group License Manager issues an appropriate message, and you can no longer execute uni-XEDIT. Contact your account representative at The Workstation Group or your local distributor when message 1005i appears.

1007e Invalid cursor position

1008e Invalid resize

1009e SET SCREEN not allowed in graphical mode

Platform-specific; cannot occur on most platforms

1010e	Invalid hex data on screen “*hexdata*”
1020e	Duplicate keybind or keybind substring “*string*”
1030e	File *filename* has been modified by another user; use FFILE/SSAVE
1040e	*command* subcommand not available in demo version
1041i	The file *filename* already exists. Do you want to overwrite that file?
1042i	The file *file* has been modified outside of the editor. Do you want to overwrite that file?
1043i	Save changes to file?
1044i	Save changes to *filename*?
1050e	READ subcommand only available with macro execution
1051e	Primary Move disabled in Input Mode
1052e	Window Manager Close cannot be used to terminate application
1060s	Save failed, file possibly damaged, copy saved as: *tempname*
1061s	Save failed, could not write temporary file: *tempname*
1062s	Save failed, file possible damaged, copy saved as: *tempname*
1063s	Save failed, could not write temporary file: *tempname*
1064s	Save failed, file possibly damaged,

1065s	no temporary copy
1066s	Save failed, directory does not have write permission: *directory-name*
1067s	copy saved as: *tempname*
1068s	Original file replaced successfully

Appendix B: System Dependencies

This appendix includes workstation- and terminal-specific information for platforms on which uni-XEDIT operates. If your particular workstation is not mentioned in this Appendix, there are no known system dependencies associated with it (or it is not supported for uni-XEDIT).

Hewlett Packard

The discussion which follows is pertinent to HP/9000 workstations running HP-UX. It applies specifically to the console keyboards. Some terminals may also exhibit the behavior described.

HP-UX provides a fairly complete terminfo definition for HP-specific keyboards. However, the following problems exist.

There is no key available to assign to the reset function since all possible keys are already mapped through terminfo. You must use the generic key sequence CTL-R.

Backtab (SHIFT-TAB) is mapped incorrectly and therefore does not function properly. This can be remedied by adding the following keybind definition to your ".profile.xedit" file:

```
“SET KEYB \ei KEY BACKTAB”
```

The sample ".profile.xedit" file on the distribution media includes this command.

The terminfo definition for the F-keys is incorrect. Each function key transmits an additional carriage return that is not included in the terminfo definition. Thus, each time you press a function key in uni-XEDIT, the effect is as if you pressed a PF key and then immediately pressed Enter.

The distribution media for uni-XEDIT contains two utility programs that reprogram the function key hardware.

- **hpfkey** – reprograms the F-keys for compatibility with uni-XEDIT (do not send the extra carriage return)
- **hpfkeyr** – resets the F-keys to their original state

HP's X-based windowing system has a feature that allows you to display a function key menu at the bottom of each window. The hpfkey program also assigns keynames for this menu. The uni-XEDIT distribution media contains two utility programs to enable and disable the menu display.

- **hpmenu** – enables the function key menu display
- **hpmenur** – disables the function menu display

The uni-XEDIT invocation script (xe) automatically invokes hpfkey and hpfkeyr if the user's TERM environment variable is set to "hpterm". It does not, however, use the keyname feature of hpfkey nor does it invoke hpmenu and hpmenur.

A sample uni-REXX program, "x," is provided to show how these utilities could be used to invoke uni-XEDIT. If you have a uni-REXX license, you may wish to use this program as it is delivered. As an alternative, you may create a shell script to execute the same steps for editor initialization and termination.

Some HP console keyboards have only eight function keys and there is no easy way to map other keys as PF9-12. The four extra, unlabelled keys near the right of some keyboards simply mimic PF5-8. Perhaps the best way to compensate for this deficiency is to redefine the first eight PF keys to perform the most desirable functions. This could be done through SET PF commands in ".profile.xedit." By using the feature of hpfkey that assigns keynames for the on-screen PF key menu and by invoking uni-XEDIT with the "x" program or a similar shell script, you can achieve a visual reminder of how you have re-defined the existing function keys.

On other keyboards, the F9-F12 keys duplicate the functionality of F1-F4. There is no way to override this behavior since the F9-F12 keys send exactly the same key sequences as F1-F4.

HP-UX does not support an internal timeout interval (the curses notimeout() function) when you login to the system across a network (rlogin, rsh, telnet). This limitation requires that keycodes to be translated by curses be received in a very short duration. Network delays often cause the characters sent by various keys (PF keys and arrow keys especially) to be delayed so that they do not meet the internal timing constraint and are not translated properly. This causes the characters sent by the key to appear on the edit screen at the location of the cursor when the key was pressed.

You may circumvent this problem by including SET KEYBIND commands for all special keys in your edit profile (".profile.xedit")

IBM Risc System 6000

The terminfo entries for the RS/6000 console keyboards (HFT terminal) provide most of the appropriate keyboard mappings for uni-XEDIT. The following are exceptions:

- the END key does not properly perform Erase-EOF
- backtab (SHIFT-TAB) does not function properly
- there is no reset key available

The following SET KEYBIND commands will overcome these deficiencies by properly maping END and

SHIFT-TAB and by mapping the PRT-SCR key to perform the reset function:

```
"SET KEYBIND \e[146q KEY ENDL"
"SET KEYBIND \e[Z KEY BACKTAB"
"SET KEYBIND \e[209q KEY RESET"
```

These commands are included in the sample edit profile provided on the distribution media, which also contains mappings for a number of the "key-pad" keys.

The console keyboard has a special key, CTRL/ACT, which does not perform any useful default function. This key is located in approximately the same position as the Enter key on a standard 3270 keyboard. Users accustomed to the 3270-style keyboard may wish to remap the Enter and CTRL/ACT keys in such a way that the "newline" and "enter" keyboard functions are in the expected positions. The following SET KEYBIND commands accomplish this:

```
"SET KEYBIND key_action KEY ENTER"
"SET KEYBIND key_enter KEY NEWLINE"
```

The IBM Risc System 6000 does not support an internal timeout interval (the curses notimeout() function) when you login to the system across a network (rlogin, rsh, telnet). This limitation requires that keycodes to be translated by curses be received in a very short duration. Network delays often cause the characters sent by various keys (PF keys and arrow keys especially) to be delayed so that they do not meet the internal timing constraint and are not translated properly. This causes the characters sent by the key to appear on the edit screen at the location of the cursor when the key was pressed.

You may circumvent this problem in one of two ways:

- include SET KEYBIND commands for all special keys in your edit profile (".profile.xedit")
- use the special AIX environment variable ESCDELAY to control the internal timing interval

In AIX, the ESCDELAY environment variable controls the internal timeout interval for key code recognition.

ESCDELAY specifies the timeout interval in units of 200 microseconds. The default value of ESCDELAY is 500, or 0.1 second. The setting required to circumvent the problem is dependent on the total network load. However, a value of 2000 (0.4 second) or 2500 (0.5 second) should be sufficient in most cases. The command to set an environment variable is dependent on your Unix shell. Select the appropriate command(s) from the table below.

Shell	**Command(s)**
C	setenv ESCDELAY *n*
Bourne	ESCDELAY=*n* export ESCDELAY
Korn	ESCDELAY=*n* export ESCDELAY

Sun Microsystems

The discussion which follows is pertinent to all workstations from Sun Microsystems. It applies to all versions of the Sun3 and the Sun4 (known variously as SLC, IPC, SPARCStation I, SPARCStation II). All system dependencies for Sun platforms are related to keyboard mappings.

The uni-XEDIT distribution media includes the file xedit.ttyswrc. This file contains the correct mappings to allow OpenLook windows to properly recognize the F- and R-keys for type 2, 3, and 4 keyboards and the F-keys for type 5 keyboards.. Copy this file to each user's HOME directory, renaming it .ttyswrc.

The .ttyswrc file is automatically executed when you create a new window. After you have located it in your HOME directory, you must start a new window for the mappings to be effective.

The uni-XEDIT distribution media also contains a sample edit profile ("profile.xedit") that includes SET KEYBIND commands to map the F-keys, R-keys, Delete (for type 2 and 3 keyboards), and Insert (for type 4 and 5 keyboards) to appropriate uni-XEDIT functions. The mappings are appropriate for the OpenLook command tool. The table below shows how the keys are mapped to uni-XEDIT keyboard functions

Key	**uni-XEDIT Functions**
F9	PF9
F10 (Type 4)	PF10
F11 (Type 4)	PF11
F12 (Type 4)	PF12
R1	PF10
R2	PF11
R3	PF12
R4	Insert
R5	Delete
R6	Reset
R7	Home
R9	PF7
R13	Erase-EOF
R15	PF8
Insert (Type 4)	Insert
Delete (Type 3 and 4)	Delete Character
Insert (Type 4 keypad)	Insert
Delete (Type 4 keypad)	Delete Character

Please note that there may be some incompatibilities between uni-XEDIT's requirements for ".ttyswrc" mappings and those of other applications. As an example:

- **For the Open Windows system**

  In some types of Open Look windows, F1 is not available to emulate PF1 since OpenLook uses F1 to display its own Help information. You can disable OpenLook's usage of F1 by commenting out the following line in your .xinitrc file:
  ```
      xmodmap -e 'keysym F1 = Help'
  ```
  To comment out a line, add a "#" to the front so that it looks like
  ```
      # xmodmap -e 'keysym F1 = Help'
  ```

If you are using an OpenLook Xterm window, the function keys (F1-F12) may not work on some systems because of a deficiency in the terminfo database entry. Use uni-KEY to define appropriate keybinds for these keys.

The cursor control arrow keys and function key F1 will not work properly with an OpenLook PSTERM window. Use uni-KEY to define appropriate keybinds for these keys.

System V/386 Workstations

The terminfo entry for "xterm", which is used with Open Desktop xterm windows and other xterm windows, does not define the F10 key in a way that is suitable for uni-XEDIT. Furthermore, there is no definition for F11 or F12.

Use SET KEYBIND commands as follows to map these keys for use with uni-XEDIT:

```
“SET KEYBIND \e[21~ PEY PF10"
“SET KEYBIND \e[23~ KEY PF11"
“SET KEYBIND \e[24~ KEY PF12"
```

The uni-XEDIT distribution media includes a sample “.profile.xedit” file that contains these definitions.

Appendix C: uni-KEY Reference

uni-KEY is a keyboard test and keybind maintenance utility from The Workstation Group. It allows you to determine what keycodes are transmitted by individual keys on your terminal keyboard. It also provides a method for defining new keybinds for use with uni-XEDIT. Keybinds defined using uni-KEY may be stored in your ".profile.xedit" file or a file of your choice.

Starting uni-KEY

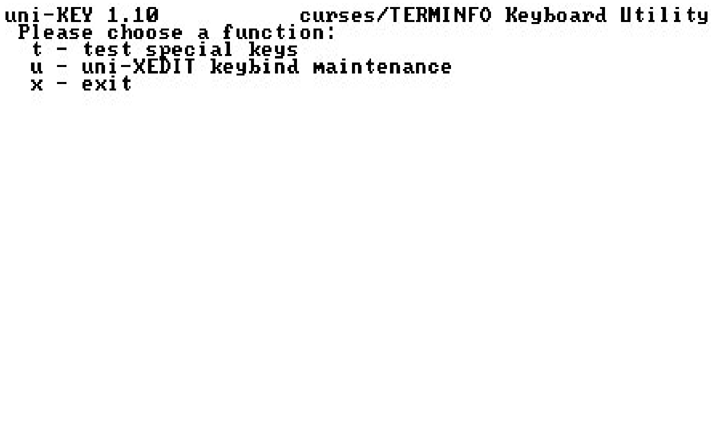
uni-KEY is installed automatically with uni-XEDIT and is executed from the Unix system prompt. The syntax of the invocation command is

key [*filename*]

Without operands, key allows you to test keys and decide later whether or not to create and save keybinds. If you choose to save keybinds, uni-KEY prompts you for the name of a file in which to capture the SET KEYBIND commands.

filename is the name of a file where uni-KEY writes keybind commands when you choose to create and save them. ***filename*** may be any Unix filename. If you want the new keyboard mappings to be effective during all uni-XEDIT sessions, the filename **must** be ".profile.xedit," and must be located in your HOME directory. If your current working directory is not your HOME directory, you **must** type the filename with its full path. If ***filename*** already exists, uni-KEY adds the new keybinds to the end of the existing file. It preserves the previous version of the file as “unikey.back.” If ***filename*** does not exist, uni-KEY creates it when you create and save any keybinds.

The uni-KEY main menu is shown below.

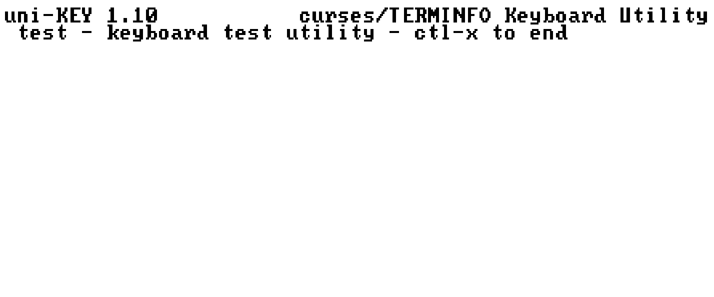


```
uni-KEY 1.10          curses/TERMINFO Keyboard Utility
 Please choose a function:
  t - test special keys
  u - uni-XEDIT keybind maintenance
  x - exit
```

From this menu you may select one of uni-KEY’s two functions or press “x” to exit the program. Once you are in the “t” or “u” functions, you may press CTL-x (when that option appears) to end that function and return to the main menu.

Testing Special Keys

To determine what keycodes are transmitted by individual keys, select “t.” The screen display changes to



```
uni-KEY 1.10          curses/TERMINFO Keyboard Utility
 test - keyboard test utility - ctl-x to end
```

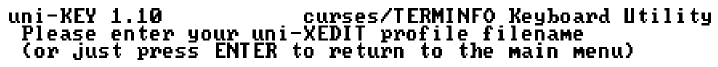
and the cursor is positioned on the first blank line of the screen. Press any key or a key combination, such as Escape followed by a key or Control in conjunction with a key. uni-KEY echoes the keycode transmitted. If the key you pressed is already mapped by curses and terminfo, that definition appears on the screen.

uni-KEY uses a timing scheme to determine when you have completed a key sequence. Allow a few seconds between key sequences. uni-KEY echoes one keycode per line on the screen. If you press keys too fast, uni-KEY cannot distinguish the end of one key sequence and the beginning of the next. When this happens, all keycodes appear on the same line until you pause briefly between key sequences.

When a key sends an unprintable character (such as Escape), uni-KEY uses the standard notation conventions to represent these codes. As examples, Escape is shown as \e and Control is shown as ^. The keycodes echoed to the screen can be used exactly as they appear in a uni-XEDIT SET KEYBIND command or to update a terminfo definition.

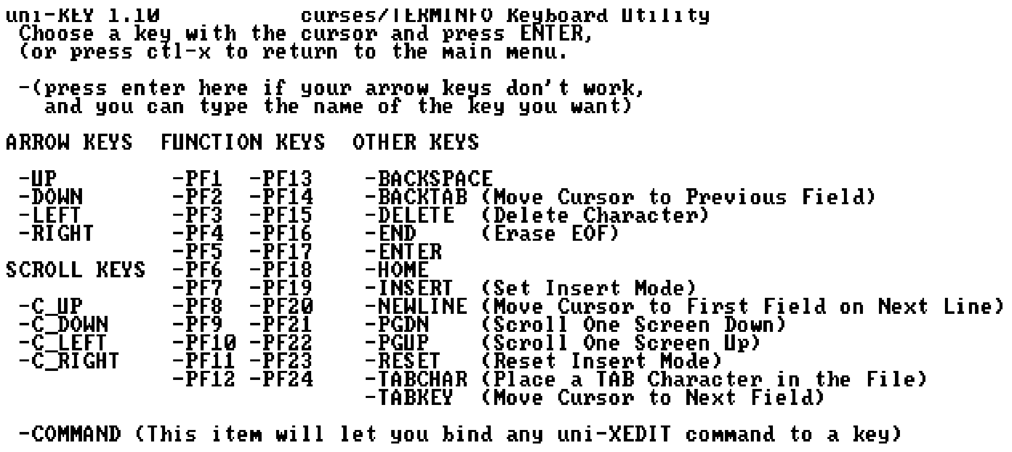
Defining Keybinds

If you select “u” from the uni-KEY main menu, the next screen display is dependent on how you invoked the program. If you did not supply a filename, the following display appears:



```
uni-KEY 1.10          curses/TERMINFO Keyboard Utility
  Please enter your uni-XEDIT profile filename
  (or just press ENTER to return to the main menu)
```

If you supply a filename at this point or if you supplied a filename when you started the program, the following display appears:



```
uni-KEY 1.10           curses/TERMINFO Keyboard Utility
 Choose a key with the cursor and press ENTER,
 (or press ctl-x to return to the main menu.

 -(press enter here if your arrow keys don't work,
   and you can type the name of the key you want)

ARROW KEYS  FUNCTION KEYS  OTHER KEYS

 -UP         -PF1  -PF13    -BACKSPACE
 -DOWN       -PF2  -PF14    -BACKTAB (Move Cursor to Previous Field)
 -LEFT       -PF3  -PF15    -DELETE  (Delete Character)
 -RIGHT      -PF4  -PF16    -END     (Erase EOF)
             -PF5  -PF17    -ENTER
SCROLL KEYS  -PF6  -PF18    -HOME
             -PF7  -PF19    -INSERT  (Set Insert Mode)
 -C_UP       -PF8  -PF20    -NEWLINE (Move Cursor to First Field on Next Line)
 -C_DOWN     -PF9  -PF21    -PGDN    (Scroll One Screen Down)
 -C_LEFT     -PF10 -PF22    -PGUP    (Scroll One Screen Up)
 -C_RIGHT    -PF11 -PF23    -RESET   (Reset Insert Mode)
             -PF12 -PF24    -TABCHAR (Place a TAB Character in the File)
                            -TABKEY  (Move Cursor to Next Field)

 -COMMAND (This item will let you bind any uni-XEDIT command to a key)
```

Use arrow keys to move the cursor to the name of the key for which you wish to create a keybind. Press Enter. uni-KEY prompts you to press the key you want to associate with this function.

uni-KEY then prompts you for the name of the key that you pressed. This is the descriptive information that identifies which keyboard key is related to this function. This information is used in a comment that documents the SET KEYBIND commands in your output file.

After you have answered all the prompts, the key name selection menu reappears so that you may define more keybinds or terminate this mode.

If your arrow keys do not function, press Enter immediately from the keyname menu. uni-KEY prompts for the name of the desired key. Type the key name as it appears on the menu (but without the leading dash) and proceed as described above.

Some systems do not support the timing feature that allows uni-KEY to determine when a complete key sequence has been received. On such systems, uni-KEY prompts you to type CTL-x after you have completed your keyboard sequence. Other prompts proceed as described above.

uni-KEY determines your current terminal type from the environment variable TERM. This terminal type is included in the comments associated with each keybind in the output file. A typical keybind generated by uni-KEY looks like:

```
"SET KEYBIND \e[209z KEY DELETE" /*sun-cmd r2*/
```

If your output file already exists, uni-KEY reads it to determine if you have an existing keybind for the keyname/key-sequence/terminal-type combination that you are now attempting to set. uni-KEY reads only those lines in the file that are in its own output format (uppercase verbs and comments that include the terminal type and key selected, as in the example above).

If a match is found, uni-KEY takes one of the following actions:

- **Key-name/Terminal-type Match:** If the key name you chose from the menu is already bound to another key for this terminal type, uni-KEY warns you of the duplication. You may delete the existing keybind and replace it with a new one, or retain the existing keybind. If you choose to retain the existing keybind, you may then bind another key to the same key name or press CTL-x to keep a single definition.
- **Key-sequence/Terminal-type Match:** If the key you pressed is already bound to a different key-name for this terminal type, uni-KEY prompts you to select another key. At the present time, there is no support in uni-KEY for removing such keybinds.

uni-KEY Limitations

This current version of uni-KEY has the following limitations:

- No more than 100 keybinds can be defined. Existing keybinds in the input file and new keybinds created in this uni-KEY session are added to determine the total.

- No more than 100 non-keybind lines from the input file will be written to the output file. Non-keybind lines are any lines that are not in uni-KEY's SET KEYBIND format described previously.
- The individual key name or uni-XEDIT command to be bound to a key sequence may not exceed 80 characters.
- The key sequence (keycode) transmitted by the individual key may not exceed 20 characters.
- A key description (for the comments) may not exceed 50 characters.
- The terminal name (from the environment variable TERM) may not exceed 20 characters.
- The fully qualified name of the file in which to capture keybinds may not exceed 256 characters.

Index

!

ALARM SOS option 164
! (exclamation point) command..175
" (Replicate) 47
"" (block replicate) 47
& (ampersand) command 176
& separator 28
.xxxx (Set symbolic name) 49
/ (Position) 48
? (question mark) command..174
| separator...28
< (Shift left)...48
<< (block shift left) 48
= (equal sign) command 173
> 49
>(Shift right) 49

A

A (Add) 42
ADD command..53
Alarm..90
ALL command...54
ALT 181
Alteration counters...181
Appending data 59
ARBCHAR 182
Arbitrary character...182
Arrow keys 15
AUTOSAVE..34-35, 179, 183

B

Backtab key 16

BACKWARD command 56
Blanks
  in strings 238
Block operations
  copy 42
  data shift 48-49
  delete 43
  exclude 46
  move 44
  replicate 47
BOTTOM command 57
Bottom of file 57

C

C (Copy) 42
CANCEL command 8, 58
CAPPEND command 59
Case 123, 170
CASE 184
Case sensitivity 30, 179, 184
CC (block copy) 42
CDELETE command 60
CFIRST command 62
CHANGE command 63
Changing data 63
CINSERT command 67
CLAST command 68
CLEAR SOS option 164
Clipboard
  See Moving text selections
CLOCATE command 69
CMDLINE 185
CMS command 72
CMSG command 73
COLOR 186
Column pointer 12
  positioning 62, 68-69
Column target
  See Targets
COMMAND command 74
Command line 12
  positioning 185

Command retrieval 174
COMPRESS command 75
Configuration settings 93, 139, 157, 167, 179
  customizing 251
  list 179
  restoring 149
  saving 134
  See also SET command options
Control characters 188
COPY command 77
Copying lines 42, 77
COVERLAY command 79
CREPLACE command 80
CTLCHAR 188
CURLINE 191
Current line 12, 48, 52, 232
  positioning 57, 109-110, 121, 126, 168, 191
curses key name
  See Keyboard mapping
curses library 17, 196
CURSOR command 81
Cursor control keys 16
CURSOR HOME 81
Cursor movement 16, 81
Customizing the editor 251
CUT command 84

D

D (Delete) 43
Data area 13
Data entry 31-33
Data integrity 35-36, 106, 153
DD (block delete) 43
Defaults
  See SET command options
DELETE command 85
Delete key 16
Deleting data 60
Deleting lines 43, 85, 137
DISPLAY 192
Documentation conventions 2-3
DOWN command 87

DUPLICAT command...88
Duplicating lines 47, 88

E

Edit keys...16-17
Edit macros...243
  argument string...256
  command set extensions 254
  contents 243
  execution...246, 248
  libraries...247
  macro name processing...257
  names 246
  pending list...257
  prefix....................................248-249, 254-257, 259-263
  profile..............5-6, 9, 30, 200, 246, 248, 250-251, 253
  sample library 254
  search order 247, 249-250
  setting XEDITPATH...250
  synonyms...257
  uni–XEDIT Extended license 243
  uses...243
Edit profile 5-6, 9, 30, 200, 246, 250-251
  examples..253
  SET options 30
Edit ring 5, 7-8, 58, 171
  split screens...226
Edit screen.......................................10, 12-13, 30, 179
  splits..226
Editing multiple files
  See Edit ring
Editor customization...251
Editor operation 5
EMSG command...90
Ending the session 34-35, 58, 140
ENTER...193
Enter key
  defining 193
Entering data 31-33
Environment variables
  ESCDELAY 278

HOME 5
PATH 114, 249
SHELL 72, 158, 175
TERM 17
TERMINFO 18
XEDITPATH 124, 247, 249
XETEMP 36-37, 107, 154

Erase–EOF key 16
Escape character 188
Excluded lines 45-46, 54, 225, 227
EXPAND command 91
Exporting data 135, 137
EXTRACT command 93
EXTRACT PENDING 259

F

F (Following) 43
FFILE command 105
FILE command 105
File names 7
changing 152, 194
wild–card characters 7, 9-10

File protection 35-36, 106, 153
File system full 36, 107, 154
FIND command 109
Finding data 109-110, 121, 240
FINDUP command 110
FNAME 194
FORWARD command 111
Full–screen display 264

G

General information 1
GET command 112

H

Help 6, 37, 114
HELP command 37, 114
HEX 195
Hexadecimal display 115, 239
HEXTYPE command 115
Home key 12, 16
HP/9000 systems 275-277

I

I (Insert) 44
IBM RS/6000 systems 277-278
INPUT command 116
Input mode....32-33, 116, 133
 split screens....226
Insert key 16
Inserting data....67
Inserting lines....42, 44, 53, 116

J

Joining lines 166

K

KEYBIND 196
Keybinds....6, 18, 196-201, 283, 285
Keyboard mapping 6, 13, 17-19, 30, 179, 196, 198-201, 283
 curses key names 197-198
 keycode sequences 197
 testing keys....284
 trouble–shooting 200-201
 uni–XEDIT keynames 199
Keycodes 196-197

L

Last command 146, 173
LEFT command 120
Limiting changes
 columns 241
 lines 218, 225
Line commands....22-23, 51
 ! (exclamation point)....175
 & (ampersand) 176
 ? (question mark)....174
 = (equal sign) 173
 ADD 53
 ALL 54
 BACKWARD....56
 BOTTOM 57
 CANCEL 8, 58
 CAPPEND 59
 CDELETE 60
 CFIRST....62
 CHANGE....63

CINSERT....67
CLAST....68
CLOCATE....69
CMS....72
CMSG....73
COMMAND....74
COMPRESS....75
COPY....77
COVERLAY....79
CREPLACE....80
CURSOR....81
CUT....84
DELETE....85
delimiters....22
DOWN....87
DUPLICAT....88
EMSG....90
EXPAND....91
EXTRACT....93
FFILE....105
FILE....105
FIND....109
FINDUP....110
FORWARD....111
GET....112
HELP....114
HEXTYPE....115
INPUT....116
LEFT....120
list....51
LOCATE....121
LOWERCAS....123
MACRO....124
MARK....125
MOVE....126
MSG....127
NEXT....128
PASTE....129
PCOPY....130
PCUT....131

PMOVE 132
POWERINP 133
PRESERVE..134
PUT 135
PUTD 137
QQUIT...140
QUERY..139
QUIT 140
READ...141
RECOVER 144
REFRESH 145
REPEAT 146
REPLACE 147
RESET 148
RESTORE 149
RGTLEFT 150
RIGHT 151
SAVE 152
SCHANGE 156
SET...157
SHELL 158
SHIFT 160
SORT 162
SOS 164
SPLTJOIN...166
SSAVE...152
STATUS 167
synonyms..234
TOP 168
UP...169
UPPERCAS...170
XEDIT 8, 171

Line numbers...211

Line target

See Targets

LINEADD SOS option..164

LINEDEL SOS option 164

LINEND...203

LOCATE command 23, 121

Logical line–end character...203

Logical line–end processing 203
Logical screen 97
LOWERCAS command 123
LRECL 205

M

M (Move) 44
MACRO command 124, 248
Macro execution 248
Macro libraries........247
  XEDITPATH 247
Macros........74
  See also Edit macros
  executing 124
  prefix........212
  search path........124
MARK command........125
MASK 206
Message line........90, 127, 207
Messages........6, 90, 127, 208, 269
MM (block move) 44
MOVE command 126
Moving lines 44, 126
Moving text selections........77, 84, 129-132
MSG command 127
MSGLINE........207
MSGMODE........208
Multiple strings........28-30

N

"Not" strings........27-28
Navigating the file 19-20
Newline key........16
NEXT command 128
Non–displayable characters........209
NONDISP 209
NULLS........210
NULLS SOS options 164
NUMBER 211

P

P (Preceding)........45
PASTE command........129

PCOPY command....130
PCUT command....131
PENDING....212
Pending prefixes....41, 148, 212, 257
PF 213
PF keys....14
  current definitions....167
  initial definitions....14, 213
  redefining....30, 179, 213
PF SOS option....164
PMOVE command....132
POINT....215
POP SOS option....164
POWERINP command....133
PREFIX....216
Prefix area....12
  display....40, 211, 216
  location....40, 216
  removing....40
Prefix commands....21-22, 39-41
  " (Replicate)....47
  "" (block replicate....47
  .xxxx (Set symbolic name)....49
  / (Position)....48
  < (Shift left)....48
  << (block shift left)....48
  >....49
  >(Shift right)....49
  A (Add)....42
  C (Copy)....42
  CC (block copy)....42
  D (Delete)....43
  DD (block delete)....43
  F (Following)....43
  I (Insert)....44
  list....39
  M (Move)....44
  matching prefixes....41
  MM (block move)....44
  P (Preceding)....45
  pending....41, 148, 212

priority 41
S (Show)..45
split screens..226
synonyms...216
TABL (Tabline) 46
uses 21
X (Exclude) 46
XX (block exclude) 46
Prefix macros..............................248-249, 254-257, 259-263
argument string 256
examples...260-263
macro name processing...257
naming 255
pending list..212, 257, 259-260
synonyms..257
PRESERVE command 134
Profile..............................5-6, 9, 30, 171, 200, 246, 250-251
examples...253
Program function keys
See PF keys
PUSH SOS option..165
PUT command...135
PUTD command 137

Q

QQUIT command 140
QUERY command 139
QUERY PENDING 258-259
QUIT command 140

R

RANGE 218
READ command 141
RECFM 219
Record format 219
Record length 205
RECOVER command...144
Redisplaying commands..174
Reformatting data 75, 91
REFRESH command 145
Reordering data 135, 137
REPEAT command 146

Repeating commands 146, 173
REPLACE command..147
Replacing data 79-80, 147
RESERVED...220
Reserved lines...188, 220, 264
RESET command 41, 148
Reset key 16
RESTORE command...149
Retrieving data 33, 112
  deleted lines 144
RGTLEFT command...150
RIGHT command..151
RS/6000 systems 277-278

S

S (Show)..45
Save 34-35, 105, 152
  automatic 30, 34-35, 179, 183
SAVE command 152
Saving data 34-35, 105, 152
SCALE 224
Scale line 12, 224
SCHANGE command 156
SCOPE 225
SCREEN 226
Screen layout 10, 12-13, 30, 179
  splits...226
Screen refresh 145
Scrolling...19-20
  horizontal 120, 150-151
  in input mode 118
  synchronized 233
  vertical...............................56-57, 87, 111, 128, 168-169
Search
  See Finding data
SELECT 222
Selecting text 125
Selective change 63, 156
SELMODE 223
SET command...23, 157, 180
SET command options..30, 179
  ALT 181

ARBCHAR....182
AUTOSAVE....183
CASE....184
CMDLINE....185
COLOR....186
CTLCHAR....188
CURLINE....191
current settings....167
DISPLAY....192
ENTER....193
FNAME....194
HEX....195
KEYBIND....196
LINEND....203
list....157, 179
LRECL....205
MASK....206
MSGLINE....207
MSGMODE....208
NONDISP....209
NULLS....210
NUMBER....211
PENDING....212, 258
PF....213
POINT....215
PREFIX....216
RANGE....218
RECFM....219
RESERVED....220
SCALE....224
SCOPE....225
SCREEN....226
SELECT....222
SELMODE....223
SHADOW....227
SHELLTERM....228
SPAN....229
SPILL....231
STAY....232
SYNCSCROLL....233

  SYNONYM 234
  TABLINE 235
  TABS 236
  TRUNC 237
  uses 30
  VARBLANK 238
  VERIFY 239
  WRAP 240
  ZONE 241
SET options 31
SHADOW 227
shadow lines 46
Shadow lines 45-46, 54, 227
SHELL command 158
Shell commands 72, 158, 175
SHELL environment variable 72, 158, 175
SHELLTERM 228
SHIFT command 160
Shifting data 48-49, 160
SORT command 162
Sort fields 162
Sort order 162
Sorting data 162
SOS command 164
SOS options
  ALARM 164
  CLEAR 164
  LINEADD 164
  LINEDEL 164
  NULLS 164
  PF 164
  POP 164
  PUSH 165
  TABB 165
  TABCMD 165
  TABCMDB 165
  TABCMDF 165
  TABF 165
SPAN 229
SPILL 231
Spilled data 49, 59, 67, 79-80, 112, 116, 147, 160, 231

Split screen 226
Splitting lines 166
SPLTJOIN command 166
    in input mode 118
SSAVE command 152
Starting uni–XEDIT 5
STATUS command 167
STAY 232
String target
    See Targets
Sun Microsystems 279-281
Symbolic names 25, 30, 49, 179, 215
Synchronized scrolling 233
SYNCSCROLL 233
SYNONYM 234
Synonyms 74
    line commands 234
    prefix commands 216
    with macros 248
Syntax diagrams 2-3
System dependencies 275
System V/386 systems 281-282

T

Tab key 15, 236
Tab stops 46, 75, 91, 235
    defining 236
TABB SOS option 165
TABCMD SOS option 165
TABCMDB SOS option 165
TABCMDF SOS option 165
TABF SOS option 165
TABKEY 16
TABL (Tabline) 46
TABLINE 235
TABS 236
Targets 23, 25-30
    preceding line commands 52
    spanning lines 229
Terminal emulators 201
terminfo database 17-19, 196, 200-201
Text selections 223

TOP command 168
Top of file 168
Trailing blanks 210
TRUNC 237
Truncation ...49, 59, 67, 79-80, 112, 116, 147, 160, 231, 237
Truncation column 237

U

Undo
  See RECOVER command
uni–KEY 18, 196, 283
  defining keybinds 285
  executing 283
  limitations 287
  testing keys 284
uni–XEDIT Extended 243
Unix commands 72, 158, 175
UP command 169
UPPERCAS command 170

V

VARBLANK 238
VERIFY 239
Virtual screen 97

W

WRAP 240

X

X (Exclude) 46
xe command 5
XEDIT command 8, 171
XEDITPATH 247, 249
  setting 250
XETEMP environment variable 36-37, 107, 154
XX (block exclude) 46

Z

ZONE 241
zone–1 column 241
zone–2 column 241