64000

HP64000 Logic Development System

Editor Manual

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environment specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Editor
Manual

LOGIC SYSTEMS DIVISION
COLORADO SPRINGS, COLORADO, U. S. A.

Printing History

Each new edition of this manual incorporates all material updated since the previous edition. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The part number on the back cover changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions. Vertical bars in a page margin indicates the location of reprint corrections.

First Printing.....October, 1979 (P/N 64980-90904)
Reprinted.....February, 1980
Reprinted.....November, 1980
Reprinted.....March, 1981
Second Edition.....November, 1981 (P/N 64980-90911)
Reprinted.....July, 1982
Third Edition.....July, 1983 (P/N 64980-90930)

Table of Contents

Table of Contents (Cont'd)

List of Illustrations

List of Tables

Chapter 1

GENERAL INFORMATION

INTRODUCTION

The 64000 System Editor is a flexible and easily used tool for creating and changing new files, and modifying existing files, by employing the directed syntax of the softkeys. In addition, block text manipulations, commands affecting multiple lines of text, and iterative command sequences can all be performed from the keyboard or from a system command file.

All editor commands can be issued from any mode. After command execution has finished, the softkey labels will be those present before command execution.

EDITOR MODES OF OPERATION

Three modes of operation make up the editor function. The modes are: command, insert, and revise.

COMMAND MODE

The editing session, when editing an existing file, will begin in the command mode. Commands issued in this mode will act with the current line of text as a reference. The current line of text is identified by an inverse video prefix, either NEW, or the line number, e.g., 88. The command mode can be entered or left at will during an editing session, through the revise or insert softkeys.

Commands, as they are issued, are retained in a memory that can be accessed by the RECALL key. The commands will be displayed in the reverse order of issue. The command can then be modified before use, or implemented by pressing the RETURN key. Although commands are stored from all modes of operation, the recall key is effective only in the command mode.

INSERT MODE

Edit sessions that generate new text files are begun in the insert mode. Two softkeys labeled *insert* are present during the edit session. The one on the left is the mode selector and is the label discussed here.

The insert mode is used to add whole lines of text to a file. Each line of text entered during the insert mode will have the prefix NEW at the left. The current line, only, will have an inverse video prefix NEW. The current line indicator, i.e., the inverse video box, will retain its position on the display, but the text will shift upward by one line each time the RETURN key is pressed.

The insert mode can be left and reentered through the softkey. The softkey label is an indicator of the mode in use. An inverse video softkey label, insert, indicates that insert mode is active. The insert mode can be exited and the command mode entered by simply pressing the insert mode softkey. The inverse video label will be replaced by the same label in standard video.

REVISE MODE

The revise mode is used to change existing lines of text. Entry to the revise mode is achieved by pressing the softkey. The label will change from standard video to inverse video, indicating that revise mode is active.

The revise mode can be exited and the command mode entered by simply pressing the revise mode softkey. The inverse video label will be replaced by the same label in standard video.

When an existing file is revised, the original file remains unchanged. By assigning the revised file to a new file name, both the original file and revised file can be retained.

KEYBOARD FUNCTIONS

The Model 64000 keyboard provides several functions applicable to the editor. The keyboard is shown in Figure 1-1.

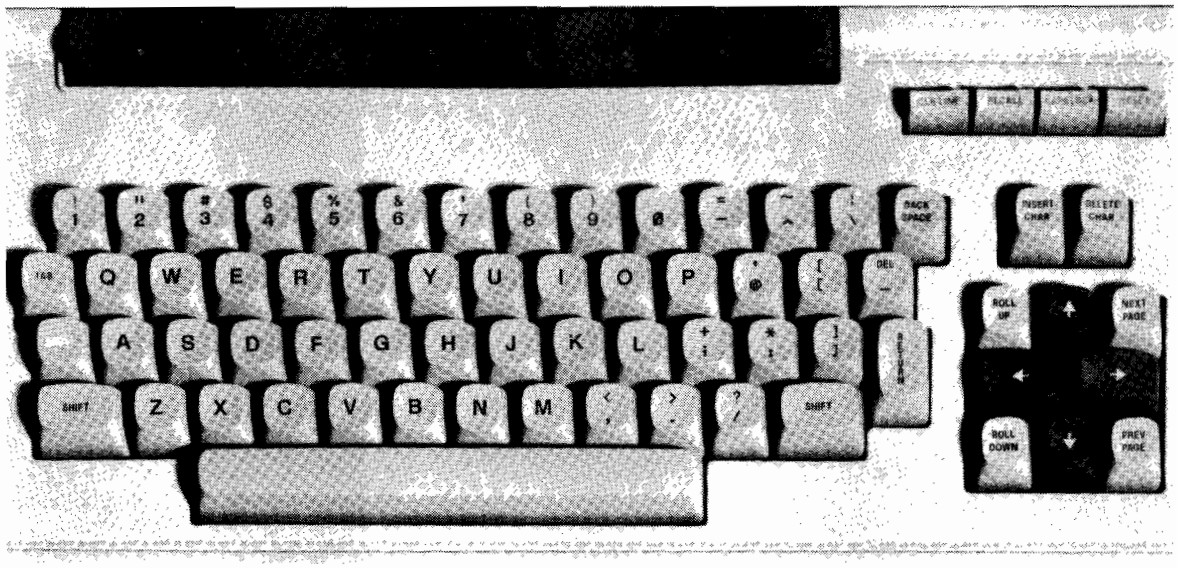

Figure 1-1. Model 64000 Keyboard

The following is a description of the functions of designated keys.

ROLL UP

The ROLL UP key moves the displayed text up one line. If the key is held down, the roll up function will repeat until the key is released or until the cursor reaches the end of the file.

ROLL DOWN

The ROLL DOWN key moves the displayed text down one line. If the key is held down, the roll down function will repeat until the key is released or until the cursor reaches the end of the file.

CURSOR CONTROLS

The left and right arrows cause the cursor to move left or right one column. If either key is held down, the function will repeat until the key is released or until the cursor reaches a boundary limit of the area. If a file contains lines wider than 80 columns, the text can be shifted with the SHIFT key and the cursor left arrow key (see below).

The up and down arrows cause the cursor to move up one line or down one line in the text display area. In the insert mode, the line being inserted will move up or down. If either arrow key is held down, the function will repeat until the key is released or until the cursor reaches the top or bottom line of the data area.

SHIFT ←

When the SHIFT key and the cursor key are held down, the text in the data area will shift to the left. The text will continue to shift until the keys are released or until column 240 of the text is displayed. Column 240 is the right limit of the display.

SHIFT →

The text can be shifted back to the right with the SHIFT key and cursor key.

NEXT PAGE

When this key is pressed, the next page, 18 lines of text, is shown in the data area of the display.

PREV PAGE

When this key is pressed, the previous page of text, 18 lines, is shown in the data area of the display.

CLR LINE

The clear line key clears the command line when the command mode is active; or in the revise mode or insert mode, clears the text line at which the cursor is positioned.

RECALL

When in the command mode, this key recalls previously executed commands from a memory stack. Each time the recall key is pressed, the previously executed command is displayed on the command line. If no commands have been issued previously, the status line will display: Recall buffer empty.

CAPS LOCK

This key locks the keyboard in all uppercase letters. When the caps lock key is pressed, the status line will display: CAPS LOCK on. When the key is pressed again, the caps lock function will be turned off and the status line will display: CAPS LOCK off. When the caps lock is on, lowercase letters can be typed by using the shift key.

RESET

Pressing the RESET key initiates a pause in system operation. A flashing inverse video: PAUSED is displayed on the status line. Pressing the RESET key a second time will abort any application module, clear the data display, and return operation to the system monitor. If after pressing the RESET key once, a decision is made not to reset the system, pressing any ASCII key (space bar recommended) will cancel the reset and continue the previous operation.

INSERT CHAR

When this key is pressed, a space with an underlined up-carat ^ is placed in the cursor location. The status line will display: INSERT CHAR on. When the insert character mode is on, a character can be entered at the place of the up-carat ^ by use of the keyboard. The up-carat will advance to the next character position when a character is entered. The up-carat prompt can also be moved with the cursor control key. The insert character function can be disabled by pressing the delete character key, the RETURN key, or the insert character key again.

DELETE CHAR

The delete chararacter key operates at the location of the cursor on the display. Each time this key is pressed, the character at the cursor location will be deleted and the text to the right of the cursor will be shifted left by one position. The delete function can be repeated by holding the delete character key down. Pressing the DELETE CHAR key will cancel the insert character function.

TAB	The TAB key causes the cursor to move to the next preselected tab stop, when in the revise mode or insert mode. In command mode, the cursor will move to the next word of the command.
SHIFT TAB	The combination of the SHIFT key and the TAB key causes the cursor to back tab to the next tab position to the left of the cursor.

DISPLAY FORMAT

The display is divided into specific sections that serve various functions of the Model 64000. Refer to Figure 1-2 for an illustration of the system display. There are four sections in the display: data area, status line, command line, and softkey label line.

The data area of the display consists of 18 lines of text, 80 columns wide. The first six columns of the display contain a prefix for the line, which is one of the following:

Line #	- a decimal number designating the sequence of the line in the original source file or its current sequence after a renumber command has been given.
NEW	- indicates the line has been added during the current edit session and has not been numbered by the renumber command.
START	- indicates the beginning of the file and is prefixed to a blank line.
END	- indicates the end of a file.
TAB	- displayed in the tabset command. Shows the new tab settings.
RANGE	- displayed in the range command. Shows the new column range.

One line on the screen is designated as the current line. Any prefix for the current line is displayed in inverse video.

The next 72 columns of the data area consist of the file text. The text may have up to 240 characters per line, although only 72 characters are displayed at any one time. The entire line can be viewed by shifting the text. Pressing the shift key and ⟵ cursor control key simultaneously will cause the text to shift from right to left. Text is automatically shifted when entering characters after column 72. Pressing the shift key and ⟶ cursor control key simultaneously will cause the text to shift from left to right. Pressing the RETURN key will cause the first 72 columns of text to return instantly to the display area.

The status line displays system status, prompt messages (instructions), error messages, and the system clock in hours and minutes (hh:mm).

The command line is composed of three lines on the display between the status line and the softkey label line. These three lines wrap around and act as one line. Commands entered through the keyboard appear in the command line. A blinking underline cursor appears in the command line as a prompt for an input. When a command or other information is input to the command line, the cursor will move to the right indicating where additional information will appear when input.

Below the command line is the softkey label line. This line displays the labels that are applicable to the top row of keys (unlabeled) on the keyboard. The right hand key is labeled ---*ETC*--- when there are additional softkey labels. When pressed, this key will provide additional labels for the softkeys. When the ---*ETC*--- label appears, the labels can be scrolled forward and will return to the first labels, after all options have been displayed.

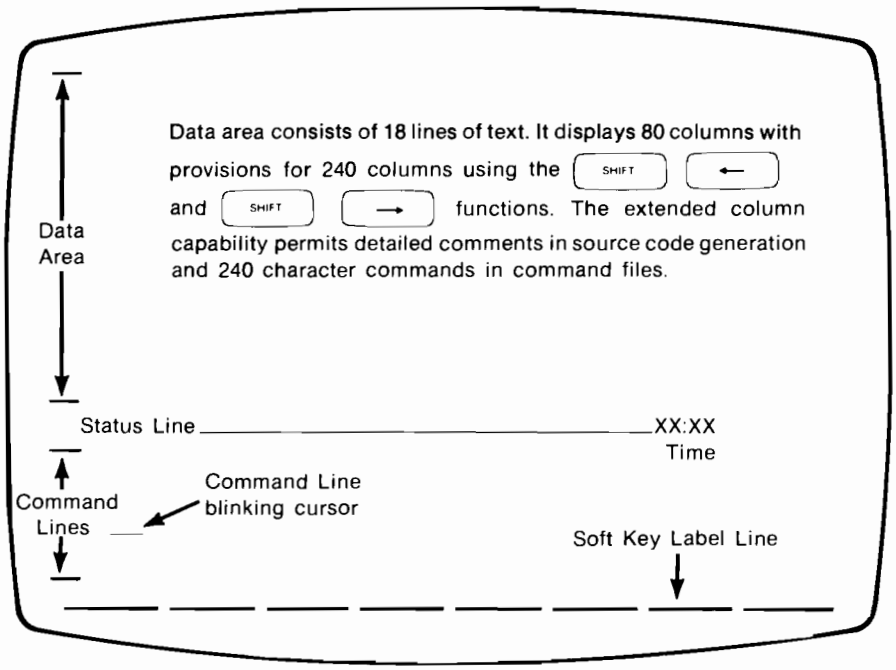

Figure 1-2. CRT Display Format

Chapter 2

EDITOR OPERATION

INTRODUCTION

An edit session is initiated by executing an edit command, either from the system monitor level or from within an edit session. A copy of the editor software to be run on the 64000 host processor is then loaded into the station RAM, and a user workspace set up in the memory.

EDIT SESSION MEMORY

The memory space available to the edit session can be viewed as two double-ended queues. These two queues share the same memory space, so when one contracts the other expands into the avalable memory. Another way to view this memory is as a single circular buffer with a display window.

EDIT SESSION FILES

SCRATCH FILES. When an edit session is started, two scratch files are created. Since more than one 64000 development station may be using copies of the editor simultaneously, the names of these files are made unique by appending the bus address of the station, namely scratch1N:HP:temp, and scratch2N:HP:temp, where N is the station's bus address. These files serve as temporary storage for text that will not fit into the memory resident in the development station.

SOURCE FILE. When the original source file is opened, enough lines to fill the display are read and placed on the display. Additional text from the source file is read into queue A. The amount of text read is limited to produce a reasonable response time. Because many edit sessions do not extend over the entire source program, and a long initial delay would be annoying, only very short files are read in their entirety before edit commands can be issued.

As various editor commands cause more of the source file to be read, the data is brought into memory and shuffled between the two double-ended queues. When the internal memory space is filled, records are written to scratch file 2 in the reverse direction. Should a command require moving to an earlier line of text, the records are written to scratch file 2 and read from scratch file 1. The original source file is never overwritten.

DESTINATION FILE. When the end command is issued, a destination file (Ntempfile:HP:temp, where N is the system bus address) is created. Text from scratch file 1, the internal buffer space, scratch file 2, and the source file is written into the destination file. The original source

file is then placed in the list of recoverable files, and the destination file renamed as source. When the scratch files are closed, they are deleted from the disc directory by the file manager.

A destination file, i.e., a new source file, can be named in one of several ways. The destination file can be named at the start of the edit session with either: 1) edit <FILE>, or 2) edit into <FILE>, or 3) edit <FILE1> into <FILE2>.

A destination file can be named at the conclusion of the edit session with either: 1) end <FILE>, or 2) save <FILE>, command.

EDITING WITH A STAND-ALONE FLOPPY-DISC BASED STATION

In order to perform edit commands on a stand-alone floppy-disc based station, the system generator module named EDITOR is required to be present at the time the edit command is executed. After the edit session is initiated, the disc containing the EDITOR module may be removed if no scratch files have been opened on it, and the editor will function properly (see special considerations below).

It is desirable to have the system generator module named FLOPPY_OP_SYS present at the time the end command is issued, in order to properly read in the monitor and its softkey labels. However, if this is not possible, the edit session will still end properly and correctly save all files.

SPECIAL CONSIDERATIONS

The editor scratch files are opened on the same disc (L.U.#) as the original source file is located. If the file is just being created, and a destination file is specified, the scratch files are opened on that disc. Otherwise the scratch files are opened on disc 0. The temporary destination file for the edit session is opened on the same disc as is specified for the final destination file. If no final destination file is specified, the temporary destination file will reside on disc 0.

To correctly end an edit session, the editor must have access to the original source file and both scratch files that have been created. In addition, there must be enough space remaining on the destination disc to accommodate the destination file. The total disc space required by the editor can be estimated to be four times the size of the original source file being edited. Thus if the source file to be edited occupies 10 pages, an additional 40 pages of disc space should be available to successfully complete an involved edit.

Given these considerations, along with the fact that there are about 250K bytes of user space available on a floppy disc, it is possible to perform successful edits of files up to approximately 80K bytes in size.

The following steps serve as a brief example of how an edit session might be structured.

1. Insert the floppy disc containing only the file to be edited into drive 0.

2. Insert the floppy disc containing the EDITOR system generator module into drive 1.

3. Issue the "edit <FILE>" command.

4. Remove the disc containing the EDITOR module from drive 1, and insert a blank formatted disc.

5. Perform the desired edit on the file.

6. Issue a "save" or "end" command, specifying disc 1 as the destination disc.

If a save or end command is issued which attempts to end onto a disc which does not have enough space remaining on it, a warning message appears on the status line stating:

ERROR: WARNING DISC A FULL DISC B NEEDED FOR RECOVERY.

Here disc A is the disc where the end attempt was made and disc B is the other disc. The command line displays the question "Is disc in drive B ready?". A formatted disc must be inserted into drive B, "yes" entered on the command line, and the save function initiated by pressing RETURN. The destination file should be successfully saved on disc B. Any part of the file that was saved onto disc A before it became full will be deleted.

If a disc becomes full while an operation which requires manipulation of a scratch file is performed, the edit session will terminate in an error condition. All scratch file errors are assumed to be fatal.

FILE ERROR MESSAGES

In the event that a file cannot be correctly read because of the way it is stored on the disc, the file is considered to be corrupt. If the editor encounters a corrupt edit or merge file, an error message to that effect will appear in the data area above the status line. If an error is discovered with a scratch file, the edit session will abort and an error message describing the situation will appear on the status line at the monitor level.

Chapter 3

EDITOR COMMANDS

INTRODUCTION

This Chapter provides general information on the use of editor commands, descriptions of the syntactical variables that make up the commands, and specific information on each command with its options.

COMMENTING EDITOR COMMANDS

Editor commands must be separated from comments by a semicolon (;) delimiter. Any text entered after a semicolon on the command line will be ignored. These entries may be used to comment in command files for clarity.

STOPPING EDITOR COMMANDS

Editor commands function on a line-by-line basis. Whenever a command that affects multiple lines is specified, a softkey labeled *stop* will be displayed as the leftmost softkey. Pressing this key during a multiple line operation stops the operation as soon as it finishes with the current line.

The *stop* softkey also appears during execution of a while statement. It will stop the execution between commands if the while statement contains an operation that can be stopped.

When executing command files, pressing the *stop* softkey will stop the execution of the command file, if possible.

Editor commands which can be stopped are listed below:

copy	list
delete	merge
extract	repeat
find	replace
<#LINE+->	retrieve

COMMAND DELIMITER FOR MULTIPLE COMMANDS

The editor accepts multiple commands on the command line through the use of the backslash (\) delimiter. Whenever a command is syntactically complete a "\" appears on the rightmost softkey. Entering a "\" causes the softkeys to return to the top level display allowing further commands to be entered. The commands save, end, edit, and <CMDFILE> may not be used in multiple commands and will not be displayed on the softkeys at this level.

---ETC---

Editor commands are shown on three lines of softkey labels.

A special softkey appears throughout the editor commands. This key is the *---ETC---* softkey. When the *---ETC---* softkey is pressed, the softkey display will change to the next line of labels. After the third line of labels is displayed, the first line of labels will return to view.

SYNTACTICAL VARIABLE DEFINITIONS

Descriptions of syntactical variables that are common to many of the editor commands are presented here in alphabetical order. The variables are:

<CMDFILE>
<COLUMN>
<COUNT>
<FILE>
<LIMIT>
<#LINE+->
<STRING>

<CMDFILE>

The <CMDFILE> variable represents the name of a source file that contains a sequence of valid 64000 system commands.

<COLUMN>

The <COLUMN> variable represents any column number in a text file. Valid entries for <COLUMN> are any positive integers from 1 thru 240.

<COUNT>

The <COUNT> variable is used as a loop counter with the while command. Valid entries for <COUNT> are positive integers from 0 thru 32,767.

<FILE>

SYNTAX

<FILE> => <FILE NAME>[:<FILE TYPE>][:<USERID>][<DISC#>]

Parameters:

<FILE NAME>	-up to nine alphanumeric characters, beginning with an upper case alphabetic character. Characters specified in excess of nine will be truncated.
<FILE TYPE>	-this represents the type of file. The file types allowed in the editor are source and listing.
<USERID>	-up to six alphanumeric characters, beginning with an upper case alphabetic character. Characters in excess of six will be truncated.
<DISC#>	-this represents the logical unit number of the system disc on which the file is located. Allowable entries are decimal numbers representing the desired disc number.

Default Values:

<FILE TYPE>	the default value for this option is dependent on the command for which <FILE> is specified.
<USERID>	the current user id.
<DISC#>	logical unit 0.

<LIMIT>

SYNTAX

```
                                  { <STRING>  }
<LIMIT>  =>  { thru  }  { <#LINE+-> }
             { until }  { start     }
                        { end       }
             { all   }
```

Parameters:

<STRING>	-any ASCII string or character. (See <STRING> description below).
<#LINE+->	-any line number in the file or signed offset from the current line.
start	-the start of text.
end	-the end of text.
all	-specifies all text in the file, starting at the first line and continuing through the last line of text.

DESCRIPTION

The <LIMIT> variable represents the limit options available in a command.

The <LIMIT> option specifies that the command is to operate from the current line of text thru, or until the limit specified, or in all of the text. This implies operation back thru text if start or a previous line of text is specified.

The keyword *thru* specifies execution up to and including the limit chosen.

The keyword *until* specifies execution up to but not including the limit chosen.

The limits for *thru* and *until* are <STRING>, <#LINE+->, start, and end.

<LIMIT> (Cont'd)

If the line number specified with this command does not exist, the line preceding the next largest line number found will become the current line when executing in ascending number direction. When executing in a descending direction, the line preceding the next smallest line number found will become the current line if the specified line number does not exist.

If the relative offset from the current line is greater than the number of lines that precede or follow the current line, the first or last line of the file becomes the current line.

<#LINE+->

The <#LINE+-> variable represents any line number in a text file or an offset from the current line. Valid entries for <#LINE+-> are any positive integers thru 32,767 or +/- 32,767 for relative line position.

<STRING>

The <STRING> variable represents any combination of ASCII characters (including control characters), the anychar symbol, [C], and/or the anystring symbol, [S]. A <STRING> must be delimited at its start and its end by the same delimiter. Valid delimiters are: quotation marks ("), apostrophes ('), and up-carets (^).

Valid string lengths vary for different editor commands. Those limitations are discussed with the specific commands.

The any character symbol, shown as a boxed C, can be entered into the command by either a softkey, *anychar*, or by pressing the control key and letter X simultaneously. When used as part of a <STRING> variable, any character, including a blank character, occupying the same relative position in the hunted string as [C], will be accepted as a valid character in the <STRING> variable.

For example, find "a[C]d" might locate the following strings: aid, a d, and, etc. The string might also be located inside a word. For example, raided, or brand, or ladder would be recognized by this find command.

The any string symbol, shown as a boxed S, can be entered into the command by either a softkey, *anystring*, or by pressing the control key and letter V simultaneously. When the any string parameter is used as part of a <STRING> variable, any string, including the null

string, occupying the same relative position in the hunted string as [S] will be accepted as a valid string in the <STRING> variable.

The anystring variable searches for the shortest string that meets the requirements. For example, find "A[S]d" would locate, in the text "All demons die.", the string "All d", and not "All demons d".

A typical search for a string, for example, find ^a[S]d^, would locate the following strings:

after the judge, aphid, almond, aged, ad, a(b + c)-d, etc.

The column number of the first character in the string will be displayed in the status line after the string is located, and the current line indicator will identify the line of text in which the string has been found.

EDITOR COMMAND SYNTAX

The syntax definitions of the editor commands are listed here in alphabetical order following the edit command definition.

Refer to the section on syntactical variable definitions for details of those variables shown within < > symbols.

edit	merge
autotab	range
<CMDFILE>	renumber
copy	repeat
delete	replace
end	retrieve
extract	revise
find	save
insert	tabset
<#LINE+->	while
list	

edit

SYNTAX

edit[<FILE1>][into <FILE2>]

Parameters:

<FILE1> - The name of an existing file being edited.

<FILE2> - The name of a destination file created from the current edit session.

Default Values:

<FILE1> -new file

<FILE2> -none

DESCRIPTION

The edit command makes it possible to create a new file, or to edit an existing file. The edit command can be issued from the system monitor or from within the edit session. If edit is called during an edit session, the current file is replaced by the new file. Changes made to the current file are not automatically saved before the new file is accessed. See the save command for details.

Syntax Examples:

edit

edit DISP

edit F1 into F2

edit into FN

edit (Cont'd)

edit - Execution of the edit command will result in the creation of a new source file. Initially, the edit session is in the insert mode and new text may be added without further command action, although all of the editor commands are usable.

edit DISP - With this command, a search is made for the named file, DISP, under the current userid. All system discs are searched in numerical order, starting with disc 0, to locate a source file having the name DISP. If a source file is not found, a further search of the discs is made for a listing file having the name DISP. If the file DISP is found, the edit session is started in the command mode.

edit F1 into F2 - With this form of the edit command, a copy of the file F1 will be brought to the display area, a destination file, F2, will be named, and the edit session begun in the command mode. The contents of F1 will remain unchanged when this edit session is ended.

edit into FN - This form of the edit command creates a new file and names a destination file, FN.

autotab

SYNTAX

autotab [column <COLUMN>]

Default Value:

<COLUMN> the last used value, or column 1 if no other value has been used.

DESCRIPTION

The autotab command turns on or off a tab function that causes the cursor to be automatically positioned as each line of text is accessed. The autotab function is independent of any tab positions.

Syntax Examples:

autotab

autotab column 37

autotab column

autotab - The autotab command turns on the mode of autotabulation based on the first non-blank character in the previous line.

If autotab is ON, the command autotab turns automatic tabulation OFF.

In the insert mode, autotab causes the cursor to be placed in the same column as the first non-blank character in the nearest preceding non-blank line.

In the revise mode, autotab causes the cursor to be placed under the first non-blank character in the current line. If the current line is blank, the cursor will be placed in the same column as the first non-blank character in the nearest preceding non-blank line.

autotab column 37 - This form of the autotab command causes the cursor to be automatically positioned at column 37. If the column number specified is larger than 72, text will be shifted to the left so that the column specified is centered in the display.

autotab (Cont'd)

autotab column - This form of the autotab command causes the cursor to be automatically positioned at the column that was last specified in an autotab column <COLUMN> command. If a value has not yet been specified during the current edit session, column 1 will be used.

<CMDFILE>

SYNTAX

<CMDFILE>

Default value:

none

DESCRIPTION

Command files are source files that contain sequences of valid 64000 system commands. Command files can be initiated from the editor in the same way that they are initiated from the monitor. In the case of command files that are started in the editor, the first command must be a valid editor command.

Command files may not be used as part of multiple command constructs in the edit session.

Syntax Examples:

```
ADDLINE
SETTAB
```

Note that there is no <CMDFILE> softkey visible among the softkey labels. When a command file is entered as an editor command, however, the softkeys will display the appropriate <PARMS> options.

copy

SYNTAX

copy [append][<LIMIT>]

Default Value:

<LIMIT> -current line of text.

Syntax Examples:

copy

copy append thru +7

copy append until "PRQ"

copy thru start

DESCRIPTION

The copy command places a copy of a line, or lines, of text (constrained by <LIMIT>) into a temporary storage buffer on disc. The original text will not be disturbed. The append option causes the copied lines to be added to existing text in the storage buffer. If append is not specified, any text in the temporary storage buffer will be overwritten.

Only one storage buffer is available, and it is shared between the copy command and extract command. The contents of the storage buffer are not protected in either case, except by the append option.

The copy command operates over the range which includes the current line of text as one boundary, and the value specified in the <LIMIT> parameter as the other boundary. Details of <LIMIT> are listed in the syntactical variable definitions, found earlier in this Chapter.

delete

SYNTAX

delete [<LIMIT>]

Default Value:

<LIMIT> - current line of text

Syntax Examples:

delete

delete until "JNZ"

delete all

delete thru 7

DESCRIPTION

The delete command removes text in the range which includes the current line of text as one boundary, and the value specified in the <LIMIT> parameter as the other boundary.

After execution of the delete command, the first line of text following the deleted text will become the current line of text.

If the <LIMIT> is not found, no text will be deleted. Details of <LIMIT> are listed in the syntactical variable definitions, found earlier in this Chapter.

end

SYNTAX

end [<FILE>]

Default Value:

<FILE> - currently specified destination file name.

Syntax Examples:

end

end CHKSUM

end INGRIN:User3

DESCRIPTION

The end command closes an edit session by placing a copy of the current work file onto the system disc. The file is entered into the directory under the name specified as the destination file name. If a file with that name already exists, that file is moved into the recoverable file list.

If a destination file is specified here, the previously specified name is automatically overwritten.

If no destination file has been named in an earlier edit command, the destination file must be specified as part of the end command.

extract

SYNTAX

extract [append][<LIMIT>]

Default Value:

<LIMIT> - current line of text.

Syntax Examples:

extract

extract until "ADD"

extract thru 26

extract append all

DESCRIPTION

The extract command removes text from the work file and places it into a temporary storage buffer on the system disc. Any text that has previously been stored will be overwritten unless the append option has been included in the command.

The append option causes the extracted text to be added to any text that is already in the temporary storage buffer.

Only one storage buffer is available, and it is shared between the extract command and copy command. The contents of the storage buffer are not protected in either case, except by the append option.

The extract command operates over the range which includes the current line of text as one boundary, and the value specified in the <LIMIT> parameter as the other boundary. Details of <LIMIT> are listed in the syntactical variable definitions, found earlier in this Chapter.

find

SYNTAX

find [<STRING>][<LIMIT>]

Default Values:

<STRING> - string specified by previous find or replace command.

<LIMIT> - start of first line after the current line thru the end of the file.

Syntax Examples:

find

find 'A[S]B'

find ^A[C]B^

find "MOV[S]" all

find "LOOP1" until 35

DESCRIPTION

The find command causes a search of the text for the occurrence of a <STRING> parameter. The search is conducted within the range of columns specified by the range command. In order for the string to be recognized, the beginning character must be found within the range.

The text lines searched with the find command are bounded by the <LIMIT> parameter on one end and by the line immediately preceding the current line or by the line immediately following the current line, depending upon whether the limit falls before or after the current line, on the other end.

For the find command, the <STRING> parameter has a maximum length of 80 characters.

Note that the <STRING> default for the find command may come from the last executed replace command. This allows repeated find and replace commands to be executed with fewer keystrokes.

insert

SYNTAX

insert <STRING>

Default Value:

None

Syntax Example:

insert "RTN"

DESCRIPTION

The insert command provides the capability of creating a new line by inserting a character string between two lines of text. The string inserted follows the current line of text.

The <STRING> parameter, for this insert command, has a maximum length of 231 characters.

Note that two softkeys labeled *insert* are present during the edit session. The one on the left is the mode selector and appears at the first line of softkeys. The *insert* label discussed here is found on the second line of softkeys, and appears above the fifth key.

<#LINE+->

SYNTAX

<#LINE+->

Default Value:

None

Syntax Examples:

6

-7

+53

0

9999

DESCRIPTION

The line number command provides the capability to specify any existing text line number, or any line at a relative (+ or -) offset, as the new current line of text.

If the line number specified with this command does not exist, the line preceding the next largest line number found will become the current line when executing in ascending number direction. When executing in a descending direction, the line preceding the next smallest line number found will become the current line if the specified line number does not exist.

If the relative offset from the current line is greater than the number of lines that precede or follow the current line, the first or last line of the file becomes the current line and a message lists the number of lines moved.

list

SYNTAX

list { <FILE> / printer } [numbered] [<LIMIT>]

Default Value:

<LIMIT> - current line of text.

Syntax Examples:

list LSTFILE

list printer numbered

list MAXINT thru 47

DESCRIPTION

The list command provides the ability to list text to a listing file named <FILE>, or to the printer, in either a numbered or unnumbered format. If the numbered format is selected and the file contains lines that are labeled with NEW instead of a number, the listing will be printed exactly as the file exists (i.e., with NEW preceding the line).

The list command operates over the range which includes the current line of text as one boundary, and the value specified in the <LIMIT> parameter as the other boundary. Details of <LIMIT> are listed in the syntactical variable definitions, found earlier in this Chapter.

merge

SYNTAX

```
merge [<FILE>] [from <#LINE+->] [thru <#LINE+->]
```

Default Values:

<FILE>	-	the current file being edited or the last previous file specified in a merge command in this edit session.
from <#LINE+->	-	first line of <FILE>
thru <#LINE+->	-	last line of <FILE>

Syntax Examples:

```
merge Sort_Proc

merge Char_LIST from 5 thru 37

merge ADDSUB:MINE from 95
```

DESCRIPTION

The merge command makes it possible to merge a file or portions of a file into the file currently being edited. Text will be added after the current line.

Only source or listing files may be merged. If the file type is not specified as part of the file name, a search will first be made for a source file and then a listing file of the specified name.

If the disc number is not indicated as part of the file name, all system discs will be searched in numerical order beginning with disc 0.

range

SYNTAX

range [<COLUMN> / all] [thru<COLUMN>]

Default Values:

<COLUMN>	-	column one
thru <COLUMN>	-	If not specified, only value of first <COLUMN> specification is used.

Syntax Examples:

range thru 200

range 25 thru 50

range 34

range

DESCRIPTION

The range command defines the range of columns of text to which all find and replace commands are restricted. Column numbers must be specified in ascending order. The range set by the range command is not applied to searches performed for <STRING> parameters defined in <LIMIT>.

If neither <COLUMN> option is specified, the present range will be shown in the display data area after command execution. To show the range, the display will separate after the current line of text and a line of R's designating the current range will appear. The cursor will be in the first column of the line, the range softkey label will be displayed in inverse video, and at this point the range specification may be changed. Changes can be made manually through the keyboard. By pressing the space bar, any R's are overwritten with spaces. The range will be set from the first non-blank character on this line to the last. Split ranges are not permitted.

If the range line is left entirely blank, the range will be set to the maximum width (240). Pressing the *range* softkey or pressing RETURN will cause the range display line to disappear and the range specification to take effect.

renumber

SYNTAX

renumber

Default Value:

None

Syntax Example:

renumber

DESCRIPTION

The renumber command is used to renumber the work file text after text has been deleted or added.

The command will renumber from the first line of text to the last line of text. There are no options or defaults for the renumber command.

repeat

SYNTAX

repeat [<#TIMES>]

Default Value:

<#TIMES> - one time.

Syntax Examples:

repeat 2

repeat

DESCRIPTION

The repeat command duplicates the current line of text one or more times and inserts the duplications immediately after the current line. The line can be duplicated up to 32,767 times.

replace

SYNTAX

replace [<STRING1>] [with <STRING2>] [<LIMIT>]

Default Values:

<STRING1>	-	string (search string) from previous replace or find command.
<STRING2>	-	string (replacement string) from previous replace command.
<LIMIT>	-	current line of text.

DESCRIPTION

The replace command operates in three distinct stages. The text is first searched, within the range and <LIMIT> specifications, for all occurrences of <STRING1>. This step is identical to that in the find command. Details of <LIMIT> are listed in the syntactical variable definitions, found earlier in this Chapter.

The beginning character of the search string, <STRING1>, must be located within the range for the string to be recognized.

The text elements found to match any [C] or [S] in <STRING1> are then substituted into their corresponding positions in <STRING2>, until all of the [C] and [S] positions are substituted, to form the replacement string.

If there are no [C] or [S] characters in <STRING1>, a blank, or a null string, is used, respectively, in <STRING2>.

If <STRING2> has more [C]'s or [S]'s than <STRING1>, the <STRING1> [C]'s [S]'s will substitute in a circular manner.

The command replace '[C][C][S]G[S]' with '[S][C][C][C][S][S]' acting on the string ABCDEFGHIJKL with the range specified to be the same column as A occupies can be used to illustrate this substitution.

replace (Cont'd)

<STRING1> may be viewed as:

$[C]_1\ [C]_2\ [S]_1\ G\ [S]_2\ J$

which then makes <STRING2> be viewed as:

$[S]_1\ [C]_1\ [C]_2\ [C]_1\ [S]_2\ [S]_1$

In this example:

$[C]_1$ = A

$[C]_2$ = B

$[S]_1$ = CDEF

G = G

$[S]_2$ = HI

J = J

KL = KL

The search string is then deleted from the text and the replacement string is inserted in its place.

Thus the text string shown will look like CDEFABABHICDEFKL after execution.

Note that the <STRING1> default for the replace command may come from the last executed find command, and the <STRING2> default may come from the previous replace command. This allows alternating use of find and replace commands without having to define <STRING1> and <STRING2> each time.

Syntax Examples:

replace ^B[S]D^ with ^DDD^

replace 'AB' with '[C]'

replace "C" with "[S]"

replace "[C][C][C]" with '[C][C][C][C]'

replace '[S]C' with ^[S][S]C^

replace ^[S]D[C][C]^ with "D[C][C][S]"

replace (Cont'd)

The text to which the above examples are applied consists of "ABCDEF". The range is only column 1.

replace "B[S]D" with "DDD" - results in no changes to the text because the search string was not found.

replace "B" with "[C]" - results in "A CDEF" because the [C] default, i.e., a blank, was substituted for B.

replace 'C' with '[S]' - results in "ABDEF" because the [S] default, i.e., the null string, was substituted for C.

replace "[C][C][C]" with "[C][C][C][C]" - This example of the replace command results in "ABCADEF" because of the [C] assignment in <STRING2>.

replace ^[S]C^ with ^[S][S]C^ - results in "ABABCDEF" because of the [S] assignment in <STRING2>.

replace ^[S]D[C][C]^ with 'D[C][C][S]' - results in text modified to "DEFABC", illustrating both [C] and [S] assignments in <STRING2>.

retrieve

SYNTAX

retrieve [<#TIMES>]

Default Value:

<#TIMES> - one time.

Syntax Examples:

retrieve 3

retrieve

DESCRIPTION

The retrieve command places a copy, or copies, of the text in the temporary storage buffer into the work file immediately following the current line of text. The last line of retrieved text then becomes the current line of text. The number of copies of text from temporary storage is specified with the <#TIMES> option. This can be any integer number. If <#TIMES> is not specified, only one copy will be retrieved.

revise

SYNTAX

revise

Default Value:

none

Syntax Example:

revise

DESCRIPTION

The revise command is utilized when changes to existing text must be made. The revise command can be used to modify an existing file, or can be used to alter a new file while it is being generated.

The revise mode can be entered and exited by use of the softkey, or exited by pressing the *insert* softkey. Pressing the *revise* softkey will position the cursor to the beginning of the current line, and the active revise mode will be noted by the inverse video label.

save

SYNTAX

```
save [<FILE>]
```

Default Value:

<FILE> - current destination file name.

Syntax Examples:

```
save TEST_RUN

save

save File:None:1
```

DESCRIPTION

Execution of the save command writes a copy of the current work file to the system disc. It is similar to the end command except that the original file is put into the recoverable file list only on the first save in the edit session. Subsequent saves do not generate recoverable files. In addition, after execution of the save command, the edit session continues rather than returning to the monitor level as with the end command.

tabset

SYNTAX

```
tabset [<COLUMN>][<COLUMN>]...
```

Default Value:

The default value is to display the current tab setting.

Syntax Examples:

```
tabset 5  10  15  20

tabset 18

tabset
```

DESCRIPTION

The tabset command sets tabs in the columns identified by number. The column numbers must be separated by a blank space. After the tabset display is initiated by pressing the RETURN key, the current tab locations will be shown by the letter T. The display will separate, as for the range command, and position the line of T's just below the current line of text. New tabs can be set at this point by placing non-blank characters in the desired columns. Existing tab settings can be removed by typing over the tab positions with a blank character, i.e., a space. Pressing RETURN again will put the tab stops into effect and clears the displayed tab line. The column 1 tabstop is always in effect.

while

SYNTAX

```
while [count <COUNT>][<multiple command>] {TEST CLAUSE}

do

[<command> [<multiple command>] {LOOP BODY}]

doend
```

Default Value:

<COUNT> one

Syntax Examples:

```
while find ^NAME^ do insert ^PLACE^ doend

while count 100 do replace ^A^ with ^B^ doend

while count 10\ find ^[C]^ do delete doend

while find ^LINE^ do extract append thru +2\find ^END^ doend
```

DESCRIPTION

The while command provides the capability to perform editor commands repeatedly, based on the result of a test. The command is structured in the form:

```
while {TEST CLAUSE}
    do
{LOOP BODY}
    doend
```

The test clause consists of one or more editor commands that are executed at the beginning of each iteration of the while command. If the result of the test clause is true, the loop body is executed. The loop body may have no commands, i.e., be empty. If the result of the test clause is false, the while command is terminated.

while (Cont'd)

The end, save, edit, and <CMDFILE> commands are not usable as part of the while command. All other editor commands are assigned truth values for use in the while command. The autotab, insert, range, and tabset commands are always true. Truth values for the remaining editor commands are assigned based on results of their execution. A summary of command truth values can be found in Appendix B.

The truth value for the test clause is true when all commands within it have true values, and false if any command within has a false value. The truth value for the while command itself is based upon the LOOP BODY only. The truth value of the LOOP BODY, like the test clause, is true if all of the commands within are true, and false if any command is false.

The keyword [count <COUNT>] is used to control the iteration of a while command. In the test clause, the keyword [count <COUNT>] evaluates as true so long as the number of iterations of the while command is less than or equal to the value of <COUNT>. If no <COUNT> is specified, it defaults to 1. In any other location the count command has no effect.

Appendix A

EDITOR STATUS MESSAGES

INTRODUCTION

The Model 64000 presents messages on the STATUS line of the CRT display. There are two types of messages displayed: syntax messages and command related messages.

Table A-1. Editor Status Messages

Command	Message	Meaning
Any command	Syntax error	The character or word above the cursor (in the command line) is syntactically incorrect.
	Corrupt file file = FILE:ID:TYPE	An error has occurred in storing the named file, and it cannot be read correctly.
	SCRATCH FILE ERROR, CORRUPT FILE	An error has occurred in storing one of the scratch files, and it cannot be read correctly.
autotab	Autotab is now on	Execution of the autotab command.
	Autotab in columnar mode, column: nnn	Execution of autotab column nnn.
	Autotab is now off	Autotab has been disabled.
copy	Copying	The copy command is executing.
	Lines copied: nn	Execution of the command is complete; the number of lines copied is indicated.
	Limit not found, no lines copied	The <LIMIT> specified with the command does not exist.

Table A-1. Editor Status Messages (Cont'd)

Command	Message	Indication
edit	File not found <FILE>	The file specified to be edited was not found.
	Do you want to lose changes? If no, File not purged, edit resumed	The current session has been changed from the original and must be saved to keep changes before editing another file.
delete	Deleting	The delete command is executing.
	Lines deleted: nn	Execution of the command is complete; the number of lines deleted is indicated.
	All text deleted, working space cleared	All text has been deleted, leaving an empty file.
	Limit not found, no lines deleted	The <LIMIT> specified does not exist.
	Stopped, lines deleted: nn	The command was stopped before completion and the number of lines deleted is indicated.
	Stopped, no lines deleted	The command was stopped before any lines were deleted.
end	FFFF already exists, delete old?	The file name specified already exists; the question "delete old?" must be answered.
	File not purged, edit resumed	This is the result of answering the question "delete old?" with a negative answer.
	No destination file	The edit file is a new unnamed file; a file name must be specified.
	Invalid input, question must be answered	This is the result of any input other than an answer to the "delete old?" question.

Table A-1. Editor Status Messages (Cont'd)

Command	Message	Indication
extract	Extracting	The extract command is executing.
	Lines extracted: nn	Execution of the command is complete; the number of lines extracted is indicated.
	Limit not found, no lines extracted	The <LIMIT> specified with the command does not exist; no lines have been extracted.
	Stopped, no lines deleted	The command was stopped before any lines were extracted.
	Stopped, lines copied: nn	The command was stopped after the lines to be extracted were copied but not removed from the file.
	Stopped, lines deleted: nn	The command was stopped after all the lines to be extracted had been copied but only the indicated number had been removed.
find	Finding	The find command is executing.
	Found in column: cc	Execution of the command is complete; the column number is indicated.
	String not found	The specified string or its default value was not found.
	Stopped, string not found	The command was stopped before the search string was found.

Table A-1. Editor Status Messages (Cont'd)

Command	Message	Indication
<#LINE+->	Positioning	The editor is positioning the file to the specified line number.
	Numbered line found	The specified line number has been found and is now the current line.
	Line not present	The specified line number is not in the file.
	Moved to line at relative offset	The line at the specified offset has been found and is now the current line.
	Line not present, minus lines offset: nnn	Offset exceeds file size; the 1st line becomes the current line. Offset: nnn indicates actual number of lines moved.
	Line not present, plus lines offset: nnn	Offset exceeds file size; the last line becomes the current line. Offset: nnn indicates actual number of lines moved.
list	Listing	The list command is executing.
	Lines listed: nn	Execution of the command is complete; the number of lines listed is indicated.
	Stopped, lines listed: nn	The command was stopped before completion and the number of lines listed is indicated.
	Stopped, no lines listed	The command was stopped before any lines were listed.

Table A-1. Editor Status Messages (Cont'd)

Command	Message	Indication
merge	Merging	The merge command is executing.
	No merge file	<FILE> was not specified and there has been no previous merge command or there is no <FILE> as specified.
	Invalid line specification	Line limits inverted or incorrectly specified.
	Lines merged: nn	Execution of the command is completed; the number of lines merged is indicated.
	Stopped, lines merged: nn	The command was stopped before completion and the number of lines merged is indicated.
	Stopped, no lines merged	The command was stopped before any lines were merged.
	File not found <FILE>	The file to be merged was not found.
range	Range is now column cc thru cc	Execution of the command is complete; the column range is indicated.
renumber	Renumbering	The renumber command is executing.
	Lines of text: nn	Execution of the command is complete; the number of lines in the file is indicated.

Table A-1. Editor Status Messages (Cont'd)

Command	Message	Indication
repeat	Repeating	The repeat command is executing.
	Lines added: nn	Execution of the command is complete; the number of lines added is indicated.
	Stopped, lines add: nn	The command was stopped before completion and the number of repeats is indicated.
	Stopped, no lines added	The command was stopped before any lines were repeated.
replace	Replacing	The replace command is executing.
	Strings changed: nn	Execution of the command is complete; the number of strings replaced is indicated.
	Limit not found, no strings replaced	The <LIMIT> specified with the command does not exist. No replacement has been made.
	Stopped, strings changed: nn	The command was stopped before completion; the number of strings changed is indicated.
	Stopped, no lines replaced	The command was stopped before any lines were replaced.
	No with replacement string specified	No string has been specified in this edit session as a "with" string for replacement.

Table A-1. Editor Status Messages (Cont'd)

Command	Message	Indication
retrieve	Retrieving	The retrieve command is executing.
	Lines retrieved: nn	Execution of the command is complete; the number of lines retrieved is indicated.
	No text has been retrieved	There is currently no text in the temporary storage buffer; no text can be retrieved.
	Recall buffer empty, no lines retrieved	There is currently no text in the temporary storage buffer; no text can be retrieved.
	Stopped, lines retrieved: nn	The command was stopped before completion and the number of lines retrieved is indicated.
	Stopped, no lines retrieved	The command was stopped before any lines were retrieved.
save	FFFF already exists, delete old?	The file name specified already exists; the question "delete old?" must be answered.
	File not purged, edit resumed	This is the result of answering the question "delete old?" with a negative answer.
	No destination file	The edit file is a new unnamed file; a file name must be specified.
	Invalid input, question must be answered	This is the result of any input other than an answer to the "delete old?" question.
while	Stack overflow, command terminated	The command stack of the while statement has been exhausted. This can only occur by specifying a large number of merges with a file name. No commands have been executed.
	While stopped, command terminated	The stop key has been pressed during the execution of a while command.

Appendix B

EDITOR COMMAND TRUTH VALUES

Command	Value
autotab	true
copy	true = lines copied false = 0 lines copied
delete	true = lines deleted false = 0 lines deleted
extract	true = lines extracted false = 0 lines extracted
find	true = string found false = string not found
insert	true
list	true = lines listed false = 0 lines listed
<#LINE+->	true = line is found false = line not found
merge	true = file merged false = no merge
range	true
renumber	true = line renumbered false = 0 lines renumbered
repeat	true = lines repeated false = 0 lines repeated

EDITOR COMMAND TRUTH VALUES (CONT'D)

Command	Value
replace	true = strings changed false = 0 strings changed
retrieve	true = lines retrieved false = 0 lines retrieved
tabset	true
while	the truth value of all the statements in the body of the do are anded cumulatively to produce the final result.

The following editor commands have no truth value and may not be used in multiple constructs:

<CMDFILE>

edit

end

save

Appendix C

EDITOR COMMANDS SYNTAX SUMMARY

EDITOR COMMANDS SYNTAX

The following is a summary of the syntax of the editor commands. The editor commands are listed in alphabetical order following the edit command.

```
edit [<FILE>1] [into <FILE>2]

autotab [column  <COLUMN>]

<CMDFILE>

copy [append][<LIMIT>]

delete [<LIMIT>]

end [<FILE>]

extract [append] [<LIMIT>]

find [<STRING>] [<LIMIT>]

insert <STRING>

<#LINE+->

list   { <FILE>  } [numbered] [<LIMIT>]
       { printer }

merge [<FILE>] [from <#LINE+->] [thru <#LINE+->]
```

EDITOR COMMANDS SYNTAX (Cont'd)

```
range  [<COLUMN>]                [thru<COLUMN>]
       [  all   ]

renumber

repeat [<#TIMES>]

replace [<STRING>] [with <STRING>] [<LIMIT>]

retrieve [<#TIMES>]

save [<FILE>]

tabset [<COLUMN>] [<COLUMN>] ...

while [count <COUNT>][<multiple command>] {TEST CLAUSE}
                        do
      [<command> [<multiple command>] {LOOP BODY}]
                       doend
```

Index

a

c

d

e

f

i

l

m

n

r

s

t

u

w

64980-90930, JULY 1983
Replaces: 64980-90911, November 1982

HEWLETT
PACKARD

PRINTED IN U.S.A.