

SOURCE EDITOR REFERENCE MANUAL

93465-90

OCTOBER 1981



Engineering Reference Number
90-93465-00G0

NAKED MINI® Division

HERTFORD HOUSE
DENHAM WAY, MAPLE CROSS
RICKMANSWORTH, HERTFORDSHIRE WD3 2XD

Telephone Rickmansworth 71211 Telex 922654

Printed in U.S.A.

REVISION HISTORY

<u>Revision</u>	<u>Issue Date</u>	<u>Comments</u>
F0	Sept. 1981	Original issue of standalone manual. Released at F0 to correspond with revision level F1 of software.
G0	Oct. 1981	Reformatting of manual with miscellaneous revisions. Adds vicinity and walkthru modes and COPY command. Corresponds to software at F300 revision level included in OS4 release G100.

Computer Automation, Inc., reserves the right to make changes in specifications at any time and without notice. The information furnished by Computer Automation, Inc., in this publication is believed to be accurate and reliable. However, no responsibility is assumed by Computer Automation, Inc., for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Computer Automation, Inc.

Computer Automation, Inc., welcomes comments and suggestions for improvements to our manuals. Should difficulty be encountered with the information in the manual or if you have a suggestion for manual improvement, please send the comments or suggestions to Computer Automation, Inc., Technical Publications, Department 11-0770, 18651 Von Karman, Irvine, California, 92713. Please provide as much information as possible to help evaluate the comment or suggestion.

PREFACE

This manual describes the use of Computer Automation, Inc's NAKED MINI® 4 Source Editor utility, which operates under the OS4 Operating System and enables a user to revise or create text in files. The edit commands and messages described in Section 4 and Appendix A also apply to the NAKED MINI® 4 Multi-Terminal Editor (MULTED). This manual can serve both as a learning tool for programmers new to the Source Editor and as a reference source for experienced users.

The OS4 System Reference Manual (93460-90) also contains information related to the use of the Source Editor Utility.

CONTENTS

SECTION 1	OVERVIEW.....	1-1
	1.0 INTRODUCTION.....	1-1
	1.1 OPERATING ENVIRONMENT.....	1-1
	1.2 DOCUMENTATION CONVENTIONS.....	1-2
SECTION 2	FUNCTIONAL DESCRIPTION.....	2-1
	2.0 INTRODUCTION.....	2-1
	2.1 INPUT TO EDIT BUFFER.....	2-2
	2.2 EDIT BUFFER.....	2-2
	2.2.1 Line Numbers.....	2-2
	2.2.2 Current Line Pointer.....	2-3
	2.2.3 Banking.....	2-3
	2.3 OUTPUT FROM THE EDIT BUFFER.....	2-4
	2.4 MODES OF OPERATION.....	2-4
SECTION 3	CALLING THE SOURCE EDITOR.....	3-1
	3.0 INTRODUCTION.....	3-1
	3.1 LOGICAL UNIT ASSIGNMENTS.....	3-1
	3.2 OS4 EDIT COMMAND.....	3-2
	3.2.1 Filenames.....	3-2
	3.2.2 Editing a Previously-Created File.....	3-3
	3.2.3 Editing One File With the Contents of a Second File.....	3-5
	3.2.4 Creating a New File From Terminal Input....	3-5
	3.2.5 Creating a New File From One or More Existing Files.....	3-6
	3.2.6 Editing Several Files at One Time.....	3-6
	3.2.7 Editing Large Files.....	3-7
SECTION 4	USING THE SOURCE EDITOR.....	4-1
	4.0 INTRODUCTION.....	4-1
	4.1 COMMAND LINE.....	4-1
	4.2 COMMAND LINE OPTIONS.....	4-2
	4.3 SET, RESET, AND LIST GLOBAL OPTIONS.....	4-3
	4.3.1 S: Suppress.....	4-4
	4.3.2 B: Blanks.....	4-4

4.3.3	C: Create.....	4-4
4.3.4	P: Print.....	4-4
4.3.5	N: No List.....	4-4
4.3.6	L: DeLimiters.....	4-4
4.3.7	V: Vicinity.....	4-5
4.3.8	W: Walkthru.....	4-5
4.4	INITIATING AN EDIT.....	4-5
4.5	EDIT BUFFER COMMANDS.....	4-6
4.5.1	ALIGN: Renumber the Edit Buffer.....	4-6
4.5.2	BANK: Save Text to be Inserted Elsewhere.....	4-7
4.5.3	COPY: Copy Text from Location to Location.....	4-7
4.5.4	DELETE: Remove Text from the Edit Buffer.....	4-8
4.5.5	E: Display Available Memory in Bytes.....	4-9
4.5.6	FIND: Locate a User Specified String in the Edit Buffer.....	4-9
4.5.7	GO TO: Reposition Current Line.....	4-10
4.5.8	INSERT: Insert Text.....	4-11
4.5.9	LIST: List Text to the System Log.....	4-12
4.5.10	MOVE: Move Text from Location to Location.....	4-13
4.5.11	P (Line Feed): Provide Spacing.....	4-14
4.5.12	PLUS (+): Reposition Current Line.....	4-14
4.5.13	SUBSTITUTE: Replace One Character String with Another.....	4-15
4.5.14	UNBANK: Insert the Bank Buffer Text into the Edit Buffer.....	4-17
4.5.15	Y-DELIMIT: Specify Delimiters for a Delimited String Search.....	4-18
4.5.16	Z: Go Back.....	4-19
4.6	EDIT BUFFER DATA TRANSFER COMMANDS.....	4-20
4.6.1	NEXT: Write Next Section of Standard Input File to Edit Buffer.....	4-20
4.6.2	READ: Copy Source Text into the Edit Buffer.....	4-20
4.6.3	WRITE: Write Text to the Ouput File (SO).....	4-21
4.7	EDIT TERMINATION COMMANDS.....	4-22
4.7.1	TERMINATE: End the Editing Process.....	4-22
4.7.2	Aborting the Edit Process.....	4-23

APPENDIXES

A	MESSAGE SUMMARY.....	A-1
B	SYNTAX SUMMARIES.....	B-1
B.0	EDIT COMMANDS.....	B-1
B.1	COMMAND LINE OPTIONS.....	B-3

FIGURES

<u>Figure</u>		<u>Page</u>
2-1	Three Phases of Source Editor Operation.....	2-1
2-2	Banking Data.....	2-4
2-3	Output From the Edit Buffer.....	2-6

TABLES

<u>Table</u>		
3-1	Source Editor Standard Logical Unit Assignments.....	3-1
4-1	Command Line Options.....	4-2

SECTION 1

OVERVIEW

1.0 INTRODUCTION

The Source Editor utility is a line oriented text editor whose basic functions are the revision of existing text and the creation of new text. The user can enter commands from a terminal or the utility can read edit commands from a file.

The operation of the Source Editor has three major phases:

- Reading lines of source text into an internal edit buffer
- Manipulating those lines in the buffer
- Writing the results of the edit to an output file.

While the text is in the edit buffer, the following operations on it are supported:

- Inserting, deleting, changing, and moving lines in the edit buffer
- Character string searching and substituting, with range control and delimiter control
- Editing large files in sections
- Merging files
- Editing a file using a previously built file containing edit commands
- Setting global options to apply to more than one command.

The utility also has an internal bank for retaining lines, with commands to transfer text between this bank and the edit buffer.

1.1 OPERATING ENVIRONMENT

The following hardware is required to execute the Source Editor:

- NAKED MINI® 4 Family computer
- 128K bytes of read/write memory
- Keyboard entry device (teletypewriter or CRT) and intelligent cable

- I/O Distributor
- Medium capacity, high capacity, or floppy disk subsystem.

The following software is required:

- NAKED MINI® OS4 Operating System
(F100 revision level or later) Diskette ID No. F46001; CAI Model
No. 93460-01

1.2 DOCUMENTATION CONVENTIONS

This manual uses the following conventions in presenting command formats:

- A backspace character deletes the previous character, so that "AA<backspace>CD" is the same as "ACD". A carriage return after a backspace deletes the entire line.
- A syntax element written in uppercase letters or special symbols other than brackets, braces, and ellipses must be entered exactly as shown.
- Square brackets [] indicate an optional term which is included or omitted at the user's discretion. A list of terms enclosed in square brackets indicates that the user can choose none or one item from the list. The brackets are not part of the input.
- Braces { } indicate a choice of elements. The user must choose one item only from the list. Braces are not part of the input.
- An ellipse ... indicates that the preceding item can be repeated one or more times. Ellipses are not part of the input.
- Lowercase letters represent user-defined parameters. These parameters must satisfy the conditions specified in the various command descriptions.

The following is a list of general user-supplied items:

pun refers to a physical unit name (e.g., DF00, TV00)

lun refers to a logical unit name (e.g., L0, SI)

device refers to a physical device, identified either by its physical unit name or by the logical unit name assigned to that physical unit name. That is, the term "device" is used in place of the construction:

{pun}
{lun}

to indicate that the user can enter either a pun or a lun at that point to identify the desired physical unit.

file refers to a file identifier. That is, the term "file" is used in place of the construction:

$$\left[\begin{array}{l} \text{pun}(/) \\ \text{lun}(.) \end{array} \right] [\text{pathname}] \text{name.extension}$$

to indicate that the user enters a filename and optional extension preceded by a pun or lun which is also optional. If the extension is omitted from a file identifier, the Source Editor searches for the version to be used in the order: .ASM, .PAS, .FTN, .BAS, .CBL, .MAC, .JCL, .TXT, .~~W~~~~W~~. If none of these versions is found, an error message is displayed.

medium refers to either a device or a file. That is, the format term "medium" is used in place of the construction:

$$\left\{ \begin{array}{l} \text{device} \\ \text{file} \end{array} \right\}$$

to indicate that the programmer can enter: (1) a pun or lun to reference a desired device, or (2) any of the specifications defined above to identify a desired file.

SECTION 2

FUNCTIONAL DESCRIPTION

2.0 INTRODUCTION

Edit operations are performed in the three phases illustrated by Figure 2-1 and listed as follows:

1. Input to the edit buffer--copying source text from the input medium to the edit buffer
2. Manipulation of lines of source text while it resides in the edit buffer
3. Output from the edit buffer--copying the edited text to the output medium.

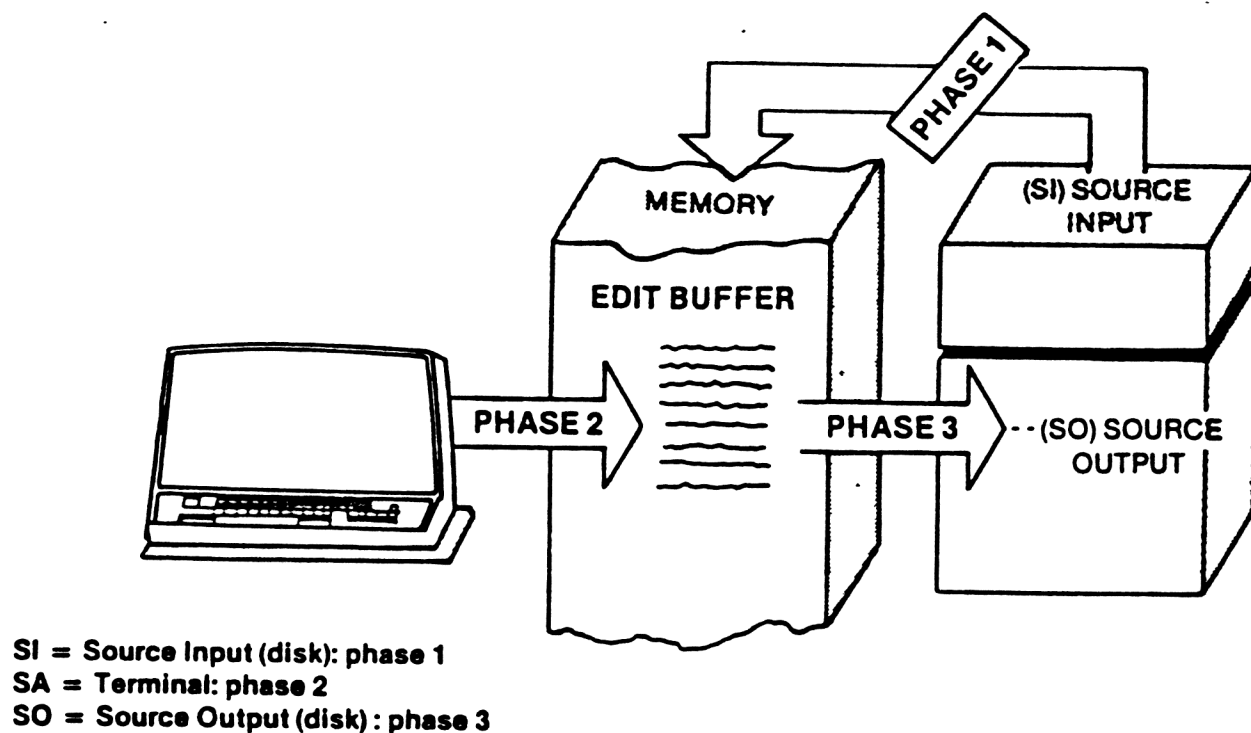


Figure 2-1. Three Phases of Source Editor Operation

The Source Editor is initiated by the OS4 EDIT command, and edit functions are performed in response to a utility prompt. When a user terminates the edit function, program control is returned to the OS4 executive.

2.1 INPUT TO EDIT BUFFER

A source file has one record per line. The user enters the READ command to transfer source records to the edit buffer. The Source Editor assigns a sequential line number to each record beginning with the first record on the source file. This means that line numbers transferred to the edit buffer are the same as the record numbers of the Standard Output (SO) file. If there are more records in the source file, the next read starts the transfer with the record following the last record read. The READ command with the G option reads records from the source file but does not enter them into the edit buffer.

The Source Editor does not allow for backward referencing in a source file, nor does it allow the user to reread records in the source file during an edit process.

2.2 EDIT BUFFER

The edit buffer is an area of memory allocated to the user. Lines from the source input are read into this buffer at the beginning of the edit and changed as necessary (line inserts, deletes, etc.). Changes performed in the edit buffer are not permanent alterations to the file until the buffer is written to the SO medium and the appropriate termination command is issued.

The same edit buffer is used when the user creates new text. The Source Editor generates an input prompt, which signals the user to enter data. The user can end this new input at any time and then use the Source Editor commands to edit the newly created data.

The size of the memory and the kind of text or language determine the number of lines that can be held in the edit buffer. For example, the edit buffer of a 128K byte machine can hold approximately 1800 lines of assembly language or 1200 lines of COBOL.

2.2.1 Line Numbers

The term line number applies only to the lines in the edit buffer, not to sequential records in the Standard Input (SI) and SO files. In the edit buffer, two or more lines can have the same line number, which means that a line number is not identical to a line. The line numbers are not part of the text itself.

Except in create mode, new line numbers are not created by editing, although line numbers can be deleted. For instance, when the source file is read into the buffer, all records and line numbers correspond. If the line delete command is used, the utility deletes both the line and the line number. The Source Editor does not keep track of line number deletions, so it cannot reassign these numbers when new lines are inserted. Instead, it assigns any inserted line the same number as the line following the insert or as the line before the insert (depending upon the command used). This can result, for example, in two or more text lines numbered 4, the second and following ones being the inserts. When lines are written to SO, they are ordered as sequen-

tial records according to the order in which they appear when listed. It is only when the file is read again to the edit buffer that the records are assigned new line numbers in ascending numerical order.

The create mode allows lines in the edit buffer to be sequentially renumbered. This is useful primarily for creating large segments of new text in an old file, where it would be exceedingly awkward for all the new lines to have the same line number. When a new file is created with the Source Editor, the input lines are numbered consecutively beginning with line number 1.

After new input has been completed, other Source Editor commands can be used to edit the text as if it came from a file. The user can specify special constructions in the edit commands that refer to the first, last, and current lines of the buffer, as follows:

- B or \$ refers to the first line in the edit buffer
- E or # refers to the last line in the edit buffer
- C or = refers to the current line number in the edit buffer.

2.2.2 Current Line Pointer

The Source Editor contains a pointer to a specific line. This specific line is called the current line and the pointer to that line is called the current line pointer. Since multiple lines can have the same line number in the edit buffer, the current line pointer is not equivalent to a line number. In a listing of the edit buffer, the current line is indicated by an equals sign (=) in front of the first character of that line. If no line numbers are specified in an EDIT command, the command references the current line by default. Otherwise, the command references the given lines and usually changes the current line pointer to reflect the command. The Source Editor also provides commands to change the current line pointer with no further action required.

2.2.3 Banking

In addition to the edit buffer, the Source Editor also contains a bank buffer, which communicates solely with the edit buffer. Text cannot be moved to the bank buffer and written to the SO medium from there. When the bank buffer accepts lines from the edit buffer, the lines become mass data. This means a user cannot return (or unbank) parts of the bank buffer. Instead, all the banked data unbanked to the edit buffer at the same time (Figure 2-2). When text is moved to the bank buffer, the previous contents of the bank buffer are discarded.

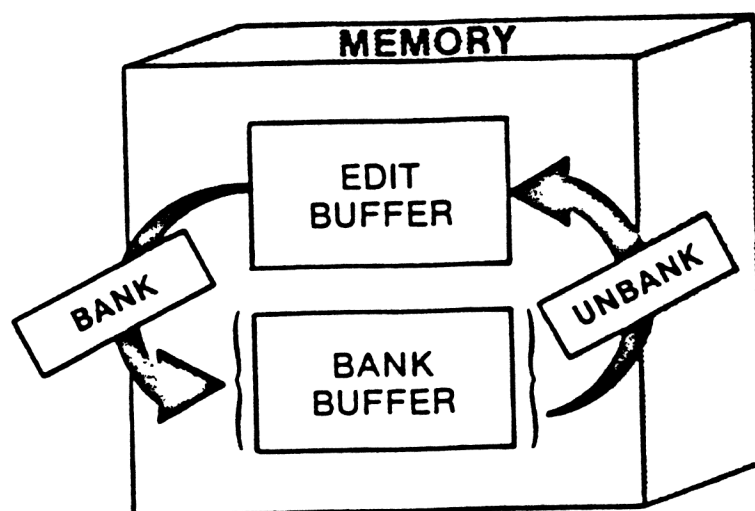


Figure 2-2. Banking Data

2.3 OUTPUT FROM THE EDIT BUFFER

When a user issues a WRITE command, the specified lines are written to the SO medium. Each line thus written becomes one record. These records are sequentially ordered, so the first line written becomes record 1, the second line written becomes record 2, and so on. It is possible, then, for line numbers in the edit buffer not to correlate with the eventual record numbers in the output file. For example, in Figure 2-3 the user read file A into the edit buffer. Line 4 was changed and immediately written to the output file. Line 2 was deleted and then lines 3 and 1 were written to the output file in that order. If the file A is again read into the edit buffer, the data appears as shown.

2.4 MODES OF OPERATION

The Source Editor supports eight global modes or command line options. When one of these modes is set, all commands that can use that option operate as if this mode were entered for each command. Each mode can be set, reset, or terminated at any time.

- Edit/Create--defines whether new text is being created or existing text is being edited.
- Delimiter--determines whether character strings are searched for with or without delimiters.
- Blank--modifies the way multiple blanks in the text are treated by the commands.
- Suppress--determines how multiple blanks are listed on the terminal and the printer.

- Print--controls whether or not the results of the commands are printed on the Listing Output (LO) medium.
- No List--defines whether or not results of the commands are listed on the terminal.
- Vicinity--causes all listing to be of the type vicinity.
- Walkthru--causes all listing, finding, and substituting to be done in the walkthru (walkthrough) mode. If both V and W are specified, the editor defaults to V.

Topic 4.2 presents further information on these options.

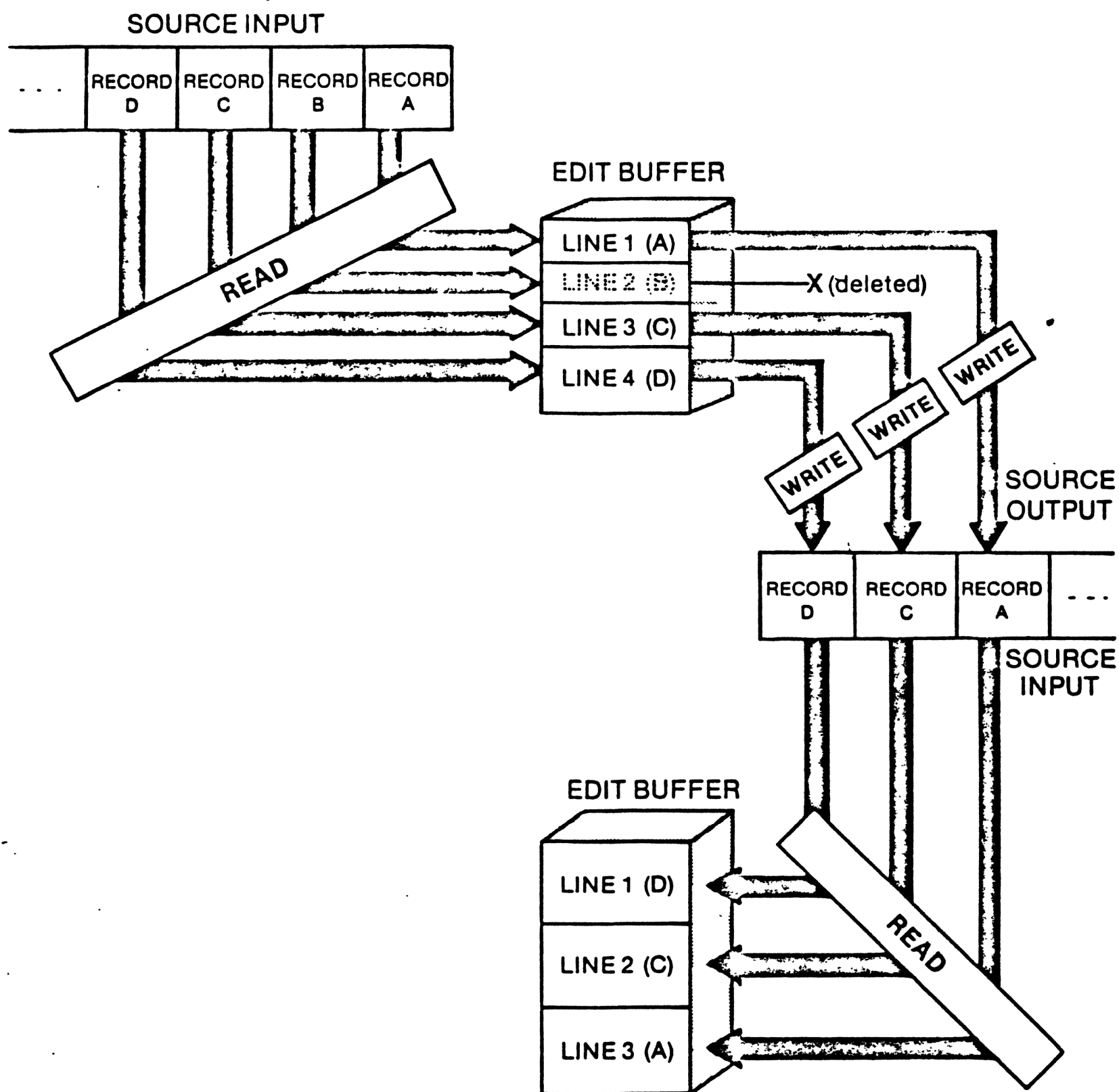


Figure 2-3. Output From the Edit Buffer

SECTION 3

CALLING THE SOURCE EDITOR

3.0 INTRODUCTION

This section presents the various formats of the OS4 EDIT command, which is used to call the Source Editor utility.

3.1 LOGICAL UNIT ASSIGNMENTS

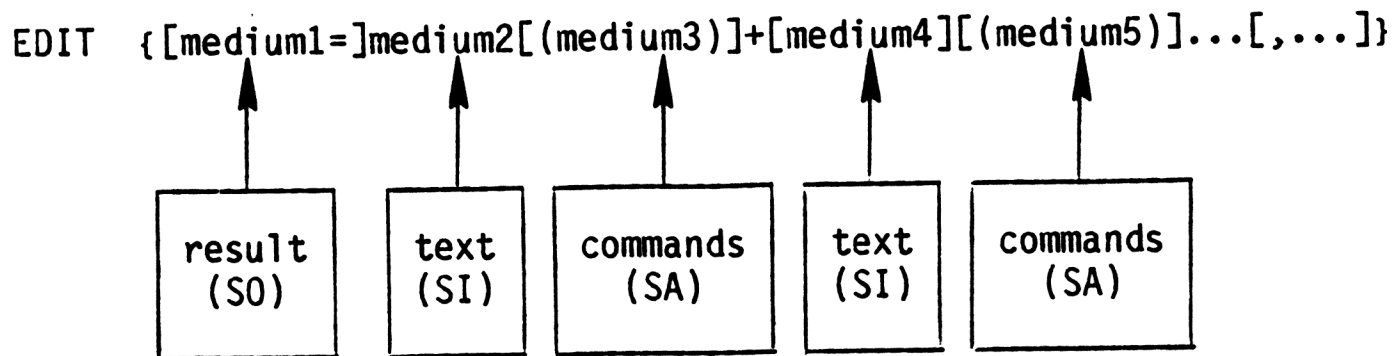
The following logical units (Table 3-1) are used by the Source Editor for its operations. Assigning logical units is discussed in the OS4 System Reference Manual.

Table 3-1. Source Editor Standard Logical Unit Assignments

LOGICAL UNIT NAME	ABBREVIATION	USE
System File	SF	Supplies loadable binary code to Source Editor
Standard Input	SI	Supplies source text
Listing Output	LO	Receives listings generated by PRINT command if the user wants a printed listing
System Log	SL	Displays output from Source Editor commands
Standard Output	SO	Receives edited text from edit buffer
Standard Alternate Input	SA	Input device that supplies edit commands to Source Editor
Command Input	CI	Supplies EDIT command line to OS4
User File	UF	Provides default location for source text that is searched when neither a lun nor a pun is given in the filename

3.2 OS4 EDIT COMMAND

The OS4 EDIT command, which calls the Source Editor, has the following format:



This command does the following:

1. Loads the Source Editor into memory
2. Reads editing commands from the specified source media (medium3, medium5, etc.)
3. Performs the requested operations on text read from the specified text source media (medium2, medium4, etc.)
4. Writes the resulting edited text to medium1.

This command source media default to the SA unit, the text source media default to the SI unit, and the destination medium defaults to the S0 unit.

If medium1 is used, the following additional rules apply:

5. If medium1 is specified without a lun and medium2 has one, medium1 is created on UF.
6. If medium1 is given without an extension and medium2 has one specified (or implied--see next topic), medium1 inherits the extension from medium2.

3.2.1 Filenames

An OS4 file identifier is composed of an optional physical or logical unit name, a period or a slash, an optional pathname, a mandatory filename of up to 11 alphanumeric characters, and an optional period and three character extension. For example:

DF1.PROG1.ASM

DH/MYDIR/SOMEFILE.FTN

If neither a pun nor a lun is included, the UF is searched for the named file. If the extension is omitted, the Source Editor searches for the version to be used, in this order: .ASM, .MAC, .BPL, .BDF, .FTN, .PAS, CBL, .XLS, .BAS, .JCL, .TXT, .DOC, .MAN, .DAT. If none of these versions is found, an error message is displayed.

The editor does not permit editing of file type devices (i.e., EDIT DMOO) or directories (i.e., EDIT main.dir). The editor responds with a string of question marks.

The user is referred to the OS4 System Reference Manual for further information. The Source Editor handles the full pathnames as discussed in that manual.

The following paragraphs present the various formats of the EDIT command and their use.

3.2.2 Editing a Previously-Created File

The Source Editor provides two formats to call the EDIT commands to edit a previously-created file.

For the first method, the Source Editor uses the default logical unit assignments:

Specified file medium = SI (input)
Terminal = SA unit (commands)
Same medium as SI unit = SO unit (destination)

For the second method, the user assigns logical units prior to the EDIT command.

3.2.2.1 Format 1: Using Default Assignments

The user inputs the EDIT command with the filename as a parameter. The Source Editor copies the specified file to the edit buffer and issues the <EDT> prompt. The format is as follows:

$\left[\begin{smallmatrix} \text{pun} \{ / \} \\ \text{lun} \{ . \} \end{smallmatrix} \right] \text{EDIT file}$

where:

pun is the physical unit on which the Source Editor is located. If neither a pun nor a lun is specified, the SF unit is searched for the Source Editor.

lun is the logical unit on which the Source Editor resides. If neither a pun nor a lun is specified, the Source Editor is read from the SF unit.

file is the file to be edited.

Example:

EDIT DF1.PROG1.ASM

In this example, DF1.PROG1.ASM is the source text file written to the edit buffer. Command input is from the terminal, and the edited file is written to DF1 and named DF1.PROG1.ASM. In addition, the old source file is renamed DF1.PROG1.BAK.

3.2.2.2 Format 2: User Provided Assignments

If the user makes specific logical unit assignments before calling the Source Editor, the EDIT command is issued without parameters.

$\left[\begin{smallmatrix} \text{pun} \{ / \} \\ \text{lun} \{ . \} \end{smallmatrix} \right] \text{EDIT}$

Example:

Assume that the programmer has issued the following ASSIGN commands to assign logical units:

```
ASSIGN SA=OC
ASSIGN SI=DF.PROG.ASM
ASSIGN SO=DF01.PROG.ASM
```

A subsequent EDIT command in the form:

EDIT

reads the Source Editor utility program from the SF unit, prompts for editing commands on the Operator Console (OC) unit, performs the requested operations on text read from the file DF.PROG.ASM, and writes the edited text to the file DF01.PROG.ASM.

3.2.2.3 Format 3: Editing Several Files with One Command

The Source Editor allows the user to edit several previously created files with one command. The first file is read into the edit buffer, after which the user then can perform any of the edit functions on it. When the terminate command is entered, the edit buffer is written to the SO unit and labeled with the first filename given in the EDIT command. The Source Editor recognizes the comma following the first filename as a signal that there is an additional file and responds with a prompt for the READ command to read the next file into the edit buffer. This process continues until all files are edited.

All files specified in the EDIT command and delimited by commas are edited separately from the others. Each receives its original filename when written to the SO unit. The filename and extension must be entered if the extension is not listed in Topic 3.2.1, or if another file exists with the same filename but with an extension that appears earlier in that list. If the filename is not found, the EDIT command aborts and control returns to OS4.

$\left[\begin{smallmatrix} \text{pun} \{ / \} \\ \text{lun} \{ . \} \end{smallmatrix} \right] \text{EDIT file1,file2,...}$

The user enters all files to be edited in the EDIT command line, separating them with commas. If the user ends the EDIT command line with a comma, the Source Editor generates the prompt (EDT...) to the CI unit when it has completed the operations specified in that command line and is ready to accept additional input.

Example:

```
EDIT DM1.PROG1,DF1.TEST.FTN,DF1.COMP.ASM
```

3.2.3 Editing One File with the Contents of a Second File

This form of the EDIT command reads editing commands from medium2, performs the requested operations on text read from medium1, and writes the resulting edited text to medium1.

```
[1un {/}] EDIT medium1(medium2)
```

where:

medium1 is the source text to be edited.

medium2 contains the edit commands used to edit medium1.

Example:

```
EDIT DF.PROG(DF1.EDM)
```

This command causes the Source Editor to perform the edit commands contained in the file DF1.EDM on the file DF.PROG.

3.2.4 Creating a New File from Terminal Input

The user can create a new file from terminal input while using the Source Editor. When the user inputs the EDIT command as shown below, an <EDT> prompt appears. The user next issues a READ command to open the edit buffer and initiate the <INP> prompt to begin input. When the user enters a line of data followed by a carriage return, the Source Editor prompts again with <INP>. To end input, the user enters a slash-asterisk (/*). At this time, the <EDT> prompt returns to signal the user that edit commands can be entered to modify the new text. At termination, the file is named with the filename and extension given in the Source Editor command.

```
[1un {/}] EDIT FILE=TV
```

where:

file is the name to be given to the text being created from the terminal.

TV indicates that the source text will be supplied from the terminal.

3.2.5 Creating a New File from One or More Existing Files

The user can also create and modify a duplicate of a file or can merge two or more files while preserving the originals.

The Source Editor reads the first file into the edit buffer. All editing of this file should be done at this time. At termination, the Source Editor writes the edit buffer to the S0 and gives the filename specified by the user. The Source Editor then checks for a plus sign in the command line. If there are additional files to be read, the Source Editor responds with a prompt for a READ command.

When the user enters a READ command, the Source Editor reads the next file into the edit buffer. The user then enters edit commands to modify the new text. Execution of edit commands on the first file is not permitted now. At termination, the Source Editor writes the file to the S0 and appends it to the previous contents of the S0. This process continues until all files are edited. The user can edit only the file currently in the edit buffer. To work with the entire merged file, the user must terminate and call the Source Editor with the new filename.

```
[pun{ / }]  
[un{ . }] EDIT file4=file1[+file2][+file3][...]
```

where:

file4 is the name of the new file created on the S0.

file1 is the name of the original file to be copied.

file2, file3 are the names of the original files to be appended to file1.

Example:

```
EDIT DF1.MERGE.ASM=DF.ONE+DF1.TWO+DM1.THREE.ASM
```

In this command, the Source Editor appends DF1.TWO and DM1.THREE.ASM to DF.ONE. The result is one file called DF1.MERGE.ASM.

3.2.6 Editing Several Files at One Time

The user can combine any number of the previous edit operations in a single EDIT command. The following format represents the complete range of possible EDIT commands.

```
EDIT [parameter[,parameter]...]
```

where:

each "parameter" defines a sequence of edit operations in the form:

[medium1=][medium2(medium3)[+medium4(medium5)]...]

Examples:

1. EDIT DF1.PROG=DF.PROG(OC)

This command causes the Source Editor to prompt for editing commands on the OC unit, perform the requested operations on text read from the file DF.PROG.ASM, and write the edited text to the file DF01.PROG.ASM.

2. EDIT DM.CON=DF.MOD1(DF.EDM)=DF.MOD2(DF.EDM)=DF.MOD3(DF.EDM)

This command performs the same editing sequence on three text files and writes the resulting edited text to a single destination file. DM.CON receives the same file extension as DF.MOD1.

3. EDIT DH.DAT.ASM(DF1.EDIT),DF1.PROG=DF.PROG(OC)

This command requests two edit operations. First, the Source Editor edits DH.DAT.ASM with the contents of DF1.EDIT and names the resulting text with the original filename. Then the Source Editor edits DF.PROG with input from the OC unit and writes the edited text to DF1 with the filename of DF1.PROG.

3.2.7 Editing Large Files

The Source Editor handles very large files in the following manner. The size of memory and the kind of text or language determine the number of lines that can be read into the edit buffer. On a READ command, the Source Editor begins to copy the file into the edit buffer, filling only 75 percent of it. If the buffer is 75 percent full before the end of file is reached, the Source Editor displays the first and last lines read in and a message indicating that the input buffer is full. The remaining 25 percent of the edit buffer is reserved to provide space for editing input. Any of the edit commands can be performed on this portion of the file.

When edit operations on the contents of the edit buffer are complete, the user enters the NEXT command. This command writes the edit buffer to the SO unit and copies the next portion of the source text from SI into 75 percent of the edit buffer. Commands entered now apply only to the text in the edit buffer. The user cannot return to the first portion of the source text during the remainder of this edit. (The UNBANK command, however, can cross the bounds of the two portions and allow the user to move a set of data from the first portion to the bank, then from the bank to the second portion.) When the last portion of a large file has been edited, the user terminates in the usual way.

SECTION 4

USING THE SOURCE EDITOR

4.0 INTRODUCTION

The Source Editor offers a broad range of commands with which the user can modify text. This section describes the command line, including its arguments, options, and delimiters. The option set, which is standard to all commands, is also presented here.

This section also presents the commands used to edit text. These commands are organized into three categories:

1. Edit buffer commands
2. Edit buffer data transfer commands
3. Termination commands.

Appendix A lists error messages and describes the conditions that cause them.

4.1 COMMAND LINE

A Source Editor command line is composed of four fields, as follows:

1. Single-character symbol for the command. I (for INSERT) and D (for DELETE) are examples.
2. String, which is used only with the FIND, SUBSTITUTE, and Y-DELIMIT commands.

Slashes are usually used to delimit the character string operands for these commands. A slash indicates the beginning of the first string. If a second string is given, a second slash separates it from the first string. An optional third slash ends the string. If other characters follow the string, however, the third slash is required. For example: S/string1/string2/W

The usual delimiter for character strings is a slash, but it can be replaced by any nonalphanumeric character used consistently within the command. If a string contains a slash, another nonalphanumeric character must be used as the delimiter.

3. Options, which further describe the original command. Each option is one character in length. Options can be specified in any order. No commas are permitted between options.

4. Range of lines affected by the command. If no range is entered, the current line is used (except in the FIND command). A comma between two numbers (e.g., 1,15) indicates the range of lines (lines 1 to 15).

These four fields of the command line are separated by commas. The command line ends with a carriage return, and only one command per line is allowed.

4.2 COMMAND LINE OPTIONS

The command line options enhance the use of the commands presented later in this section. These options are not standard to all commands, but each option operates in a similar manner for every command that uses that option. Applicable options are listed as parameters in command lines.

Some of the options listed in Table 4-1 are global and can be set globally by the O command or locally in individual commands. When these options are used locally, they temporarily override the default values of the global options. G in the table indicates that an option is both local and global.

Table 4-1. Command Line Options (Page 1 of 2)

OPTION	GLOBAL	MEANING	RELEVANT COMMANDS
A		All occurrences within range of lines	FIND
B	<u>G</u>	Multiple blanks seen as single blanks	
C	<u>G</u>	Create	
D		Destroy	UNBANK
D		Delete delimiters	Y-DELIMIT
D		Delete to end of line	SUBSTITUTE
F		First occurrence in line only	SUBSTITUTE
G		Garbage--records are destroyed	READ, TERMINATE
G		Wildcard	FIND, SUBSTITUTE
I		Immediate insert	INSERT, MOVE, UNBANK
I		Insert delimiters	Y-DELIMIT
K		Keep in edit buffer (else purge)	BANK, WRITE
K		Keep in bank buffer (else purge)	UNBANK

Table 4-1. Command Line Options (Page 2 of 2)

OPTION	GLOBAL	MEANING	RELEVANT COMMANDS
L	<u>G</u>	Delimited string	DELETE
N	<u>G</u>	No listing of result of any edit command	
P	<u>G</u>	Print on the L0 device	
R		Replace	
S	<u>G</u>	Suppress expansion of multiple blanks	LIST, FIND, SUBSTITUTE
V		Vicinity (11 previous lines to 11 following lines)	
W		Walkthru	
X		Extend (add to end of line)	

4.3 SET, RESET, AND LIST GLOBAL OPTIONS

When a global option is set, all commands that can use that option default to the option. Any of the eight options can be set (or reset) at one time.

0 is the universal option command, which lists the currently set modes of operation as described in Section 2. If the user enters another option after the 0, that option is set and listed. The options cannot be entered without the preceding 0. Q resets the listed global options.

Format 1: 0[S][B][C][P][N][L][V][W]

Format 2: Q[S][B][C][P][N][L][V][W]

where:

0 = lists the currently set global options only.

Q = resets the listed global options.

options = as described in Topics 4.3.1 through 4.3.8.

The following paragraphs present the individual options.

4.3.1 S: Suppress

Suppress multiple blanks when listing or printing. When this option is set, a group of blanks is listed as one blank.

Format: OS = lists current global options and sets suppress mode.

4.3.2 B: Blanks

When the blank option is set, multiple blanks are treated as single blanks. Thus, on the FIND and SUBSTITUTE commands, any number of blanks match any other number of blanks. For example, one or two blanks are seen as matches for five blanks.

Format: OB = lists current global options and sets blank mode.

Example: F/ABCØD/B = find all occurrences of "ABC", followed by one or more blanks, followed by "D".

4.3.3 C: Create

When create mode is set, the user can align or renumber lines in the edit buffer relative to line 1. If an INSERT is executed in create mode, the Source Editor updates the line numbers of the edit buffer in relation to line one. (Normally, no new line numbers are created on INSERT.) See the ALIGN command for additional information.

Format: OC = lists current global options and sets create mode.

4.3.4 P: Print

When the print global option is set, the results of execution of commands with that option are printed on the LO device.

Format: OP = lists current global options and sets print mode.

4.3.5 N: No List

When no list is set, listings to the SL are suppressed. The default condition is that the result of each command with the option is listed on the SL, which is usually the terminal.

Format: ON = lists current global options and sets no list mode.

4.3.6 L: DeLimiters

Characters can be defined with the Y-DELIMIT command to be valid or invalid delimiters for character string matches. When the limiters mode is set, the

Table 4-1. Command Line Options (Page 2 of 2)

OPTION	GLOBAL	MEANING	RELEVANT COMMANDS
L	<u>G</u>	Delimited string	DELETE
N	<u>G</u>	No listing of result of any edit command	
P	<u>G</u>	Print on the LO device	
R		Replace	
S	<u>G</u>	Suppress expansion of multiple blanks	LIST, FIND, SUBSTITUTE
V		Vicinity (11 previous lines to 11 following lines)	
W		Walkthru	
X		Extend (add to end of line)	SUBSTITUTE

4.3 SET, RESET, AND LIST GLOBAL OPTIONS

When a global option is set, all commands that can use that option default to the option. Any of the eight options can be set (or reset) at one time.

O is the universal option command, which lists the currently set modes of operation as described in Section 2. If the user enters another option after the O, that option is set and listed. The options cannot be entered without the preceding O. Q resets the listed global options.

Format 1: O[S][B][C][P][N][L][V][W]

Format 2: Q[S][B][C][P][N][L][V][W]

where:

O = lists the currently set global options only.

Q = resets the listed global options.

options = as described in Topics 4.3.1 through 4.3.8.

The following paragraphs present the individual options.

4.3.1 S: Suppress

Suppress multiple blanks when listing or printing. When this option is set, a group of blanks is listed as one blank.

Format: OS = lists current global options and sets suppress mode.

4.3.2 B: Blanks

When the blank option is set, multiple blanks are treated as single blanks. Thus, on the FIND and SUBSTITUTE commands, any number of blanks match any other number of blanks. For example, one or two blanks are seen as matches for five blanks.

Format: OB = lists current global options and sets blank mode.

Example: F/ABCØD/B = find all occurrences of "ABC", followed by one or more blanks, followed by "D".

4.3.3 C: Create

When create mode is set, the user can align or renumber lines in the edit buffer relative to line 1. If an INSERT is executed in create mode, the Source Editor updates the line numbers of the edit buffer in relation to line one. (Normally, no new line numbers are created on INSERT.) See the ALIGN command for additional information.

Format: OC = lists current global options and sets create mode.

4.3.4 P: Print

When the print global option is set, the results of execution of commands, with that option are printed on the LO device.

Format: OP = lists current global options and sets print mode.

4.3.5 N: No List

When no list is set, listings to the SL are suppressed. The default condition is that the result of each command with the option is listed on the SL, which is usually the terminal.

Format: ON = lists current global options and sets no list mode.

4.3.6 L: DeLimiters

Characters can be defined with the Y-DELIMIT command to be valid or invalid delimiters for character string matches. When the limiters mode is set, the

delimiter buffer is examined. One valid character must bind the front of the string, and another valid character must bind the end of the string to be considered a match. (In the default mode, commands examine character strings independent of their surroundings.)

Format: OL = lists current global options and sets delimiter mode.

4.3.7 V: Vicinity

When vicinity is set, FIND, SUBSTITUTE, LIST, and GO TO do a vicinity listing of the current line and ± 11 lines.

Format: OV = lists current global options and sets vicinity mode.

4.3.8 W: Walkthru

When walkthru is set, FIND, SUBSTITUTE, LIST, and GO TO do a walkthrough listing (see LIST).

Format: OW = list current global options and sets walkthru mode.

NOTE

The symbols A, B, C, and E have different meanings in the range field of the command and in the options field. In the range field:

A = all lines; equivalent to "B,E"
B = beginning line
C = current line
E = end line.

To avoid confusion, a user can set off the range field with a comma. For example:

S/ØA/A/B,23	SUBSTITUTE (using the blank option) in line 23.
-------------	---

S/ØA/A/,B,23	SUBSTITUTE in lines from the beginning through line 23.
--------------	---

4.4 INITIATING AN EDIT

The Source Editor is loaded by the EDIT command line to OS4. After loading, a program announcement appears on the CRT followed by the message "outputfile=inputfile," where the "outputfile" refers to the destination medium of the edited source text and "inputfile" refers to the original source text as indicated by the EDIT command. When only one file is given, the output file

assumes the .EDT extension and is automatically renamed to the input filename upon normal termination (T).

After the message, the prompt "<EDT>" appears, signaling the user that the utility is ready to accept Source Editor commands.

In most cases, the first command entered in edit is the READ command, which copies source file records from the SI into the edit buffer. Without this command or the INSERT command, the buffer is empty. The user who tries to execute commands on nonexistent lines is prompted with an "end-of-data" message. If the TERMINATE command is entered before READ or INSERT, the edit is terminated with the resulting file unchanged.

After READ has been executed, the Source Editor displays the first and last lines read into the edit buffer and another <EDT> prompt, indicating that it is ready for the next command.

An exception to the above procedure occurs when a new file is created from terminal input. The READ command displays an <INP> prompt asking the user to input data. Input is terminated when /* is entered in response to the <INP> prompt. Otherwise, the Source Editor functions as explained above.

4.5 EDIT BUFFER COMMANDS

The following commands are used to manipulate data in the edit buffer, change the current line pointer, locate character strings within the buffer, generate listings, and so on. Each command is entered as a single letter only. The commands are listed in alphabetical order.

4.5.1 ALIGN: Renumber the Edit Buffer

This command renumbers the lines of text in the edit buffer sequentially starting with line number 1. The current line pointer is not affected by the command.

The ALIGN command is provided for the user who has added several lines in a file and wants to work with an edit buffer that is numbered sequentially, rather than having all the new lines with the same number.

Format: A

NOTE

If the user returns to the edit mode after being in create mode, the new line numbers do not change. The appropriate measures should be taken before doing a read.

4.5.2 BANK: Save Text to be Inserted Elsewhere

The user can save, or bank, a line or a group of contiguous lines of text from the edit buffer. (Another command, UNBANK, inserts the banked lines at a specified location in the edit buffer.) BANK removes lines from their place in the edit buffer and stores them in a bank buffer, or copies lines to the edit buffer rather than removing them (the keep option). The bank buffer holds data through a READ, WRITE, or NEXT command, but it is not retained after a TERMINATE command is given.

If the keep option is used, the current line number is not changed. Otherwise, the current line becomes the line number of the line following the last line moved or copied.

Format: B[K][,line1[,line2]]

where:

- K keeps option banks specified lines while also keeping them in their original location.
- line1 used alone, refers to all lines having that line number.
- line2 used with line1, refers to a range of records from line1 through line2. If neither line1 nor line2 is specified, the range defaults to the current line.

Examples:

1. B = bank the current line and do not keep (move line to the edit buffer).
2. BK6 = bank lines with line number 6 and also keep in original location (copy lines to the edit buffer).
3. B6,12 = bank lines from line 6 to line 12.

4.5.3 COPY: Copy Text from Location to Location

This command copies lines from one section of the edit buffer to another without using the bank buffer. The original lines are left intact. If an attempt is made to copy into the middle of the block of lines already being copied, an error will result.

If several lines share the destination line number, the copy is done before the first, or after the last, line having the specified line number. The I option means insert before the specified line number. The current line becomes the first line copied.

Format: C[I][N][[line1,[line2,]]dline]

where:

- I inserts the lines before, rather than after, the destination lines.
- N does not list result. //
- line1 when only two line numbers are used, line1 refers to the lines to be moved to follow dline.
- line2 when three line numbers are used, line1 and line2 refer to a range of line numbers to be moved to follow dline.
- dline destination line: when only the destination line number is used, the current line is moved to follow dline. If dline is not specified, the current line is duplicated.

Examples:

1. C = duplicate the current line.
2. C64 = copy the current line to follow lines numbered 64.
3. C32,64 = copy lines numbered 32 to follow lines numbered 64.
4. CI32,38,64 = copy line numbers 32 through 38 to the position before lines numbered 64. (The I option applies only to the destination.)

4.5.4 DELETE: Remove Text from the Edit Buffer

The DELETE command erases from the edit buffer the line or lines specified in the command, or the current line if no line number is given, and lists the next (current) line.

The R option allows the user to delete text and replace it with input data in one operation. When this option is used, an insert prompt <INS> is displayed after the specified lines have been deleted. The user can then input data following the procedures given for the INSERT command. With the R option, the last line of input becomes the current line. Otherwise, after execution of the DELETE command, the line following the last deleted line becomes the current line. All inserted lines are given the line number of the preceding line in the edit buffer.

Format: D[N][P][R][,line1[,line2]]

where:

- N suppresses a listing of the new current line.
- P prints on the LO device.
- R is the replace option.
- line1 used alone, refers to all records having that line number.

line2 used with line1, refers to a range of records from line1 through line2.

This format defaults to deleting the current line.

Examples:

1. DN = delete current line, no listing.
2. D140 = delete all lines with line number 140, listing the first following line.
3. DR140,150 = delete all lines with line number 140 through line number 150, replacing them with user input.

4.5.5 E: Display Available Memory in Bytes

The E command displays the currently available unused area in the edit buffer in bytes. The current line is not affected.

Format: E

4.5.6 FIND: Locate a User Specified String in the Edit Buffer

The FIND command searches the edit buffer in the given range of lines for the string of characters specified in the command. Only the first occurrence of the string is found unless the A or W option is used. If no range is given, FIND starts searching at the current line +1.

Format: F/string/[W][B][G][L][N][P][S][V]^A[,line1[,line2]]

where:

/string/ is the character string desired. The string is enclosed in slashes, but the last slash is optional if no other parameters are on the line. If the string to be found contains a slash, any other nonalphanumeric character that does not exist in the string can be used instead.

W provides a "walkthru." The first line in the range that contains the string is displayed, then the prompt (XXX) is given. The user can input either an X or a carriage return. Entering an X causes the FIND to stop. The current line pointer is set to the line found. A carriage return causes the next line containing the string to be displayed together with the prompt (XXX). If no more lines are found, the current line pointer is not affected by the command.

B multiple blanks are seen as single blank.

- G wildcard: @ matches any single character.
- L allows the user to specify that preset alphanumeric characters must bound the matching string of characters. The user also can specify that preset nonalphanumeric characters must not bound the matching string. The characters are preset with the Y-DELIMIT command.
- N does not list result.
- P prints on the L0 device.
- S suppresses blanks on listing.
- V lists vicinity around the found line.
- A finds all occurrences of the string in the range and displays them. This option does not change the current line pointer.
- For the default, search stops on the first line in the specified range which contains the string. No other lines in the range are searched. The current line pointer is set to that line.
- line1 used alone, refers to all records having that line number.
- line2 used with line1, refers to a range of records from line1 through line2.

If neither line1 nor line2 is specified, the format defaults to searching the entire edit buffer.

Examples:

1. F/123 = find the first occurrence of the character string "123" in the entire edit buffer.
2. F/123/AP5,10 = find all occurrences of the string "123" in the range of line number 5 through line number 10, printing them on the L0.
3. F/123/L,10,100 = find the first occurrence of the character string "123," checking for delimiters, from line number 10 through line number 100.

4.5.7 GO TO: Reposition Current Line

The GO TO command repositions the current line pointer to the line number given or advances the current line pointer one line if no line number is given. A carriage return after the <EDT> prompt also advances the current line pointer one line. If there is more than one line with the given line number, the pointer is positioned to the first one.

Format: G[N][P][S][V][,line1]

where:

N does not list results.

P prints on the LO device.

S suppresses blanks on listing.

V lists the vicinity of line1 and repositions the current line pointer.

line1 refers to line specified by the user. If no line number is specified, the format defaults to advancing current line pointer to next line (same as carriage return).

Examples:

1. G = advance current line pointer to the next line.
2. GN8 = reposition current line to line 8 with no listing.
3. GV16 = reposition current line to line 16 and list vicinity.
4. Carriage return = advance current line pointer to the next line.

4.5.8 INSERT: Insert Text

The user can insert text before and after a given line number, or if no line number is given, before and after the current line. When this command is given, the Source Editor displays an <INS> prompt indicating that the user can input data. A carriage return ends each line of input and displays a new <INS> prompt for continued entry. Input is terminated by a line consisting of a carriage return only. If more than one existing line share the given line number, the insertion occurs after the last line of the given line number.

If the INSERT command is entered before any READ commands, the Source Editor responds with an <INS> prompt. All data entered at this point has the line number 0. A READ command following the INSERT command appends the text from the SI to the end of the inserted data and numbers all lines sequentially, starting with 1, 2, etc.

In general, inserted lines to an empty buffer are all assigned the line numbers of the next line to be read from the SI unit minus 1.

The I option with this command inserts data immediately in front of the given target line or lines. All inserted lines are given the line number of the target line, if it exists; otherwise, the lines are given the line number of the line directly above or below it. The last line of input becomes the current line. If input is terminated immediately after entering insert mode,

the current line is set to the first line following the point at which insertion would have occurred.

A question mark (?) line only, entered at any time during input displays a template which numbers the columns across the terminal screen. These numbers aid in entering characters at specific column positions.

Format: I[I][line1]

where:

- I inserts line in front of the target lines.
- line1 refers to the lines specified by the user; if there is more than one line with the same number, all those lines become target lines. If line1 is not specified, then the current line becomes the target line.

Examples:

1. I = insert lines after the current record.
2. II,10 = insert lines immediately before line 10.

4.5.9 LIST: List Text to the System Log

The LIST command lists specified lines of the edit buffer to the SL unit, which is usually assigned to the terminal. If no line number is given, the current line is listed. The range of lines to be listed can be given in the command. LIST does not change the current line pointer.

Format: L $\begin{bmatrix} V \\ W \end{bmatrix}$ [P][S] $\begin{bmatrix} A \\ \text{line1[,line2]} \end{bmatrix}$

where:

- V causes a listing of 11 lines prior to the current line, the current line, and 11 lines beyond the current line.
- W walks through the range of listed lines. A prompt <FXR> gives the user the option to:
 - F = display the next page (forward)
 - X = terminate listing
 - R = display the previous page (reverse).
- P prints on the L0 device.
- S suppresses multiple blanks in a listing. (All multiple blanks are printed as one blank.)
- A lists all lines in the edit buffer.

line1 used alone, refers to all lines having that line number.

line2 used with line1, refers to a range of lines from line1 through line2.

This command line defaults to listing the current line only.

Examples:

1. L = list the current line.
2. LP150 = list all lines numbered 150 and print them on the L0.
3. LS150,180 = list line number 150 through 180, suppressing multiple blanks (print only one blank for multiple blanks).

Option combinations:

LV = list the vicinity: the 11 lines previous to and the 11 lines following the current line.

L,B,50 = list all lines from the beginning of the edit buffer through lines numbered 50.

4.5.10 MOVE: Move Text from Location to Location

The MOVE command performs the bank and unbank operations in one command: the specified text is removed from its original location in the edit buffer, stored in the bank buffer, and inserted into the given location. This command differs from BANK in that the original occurrence of the lines cannot be saved and the lines can be inserted only in one location. The bank buffer is emptied upon insertion into the new location.

If several lines share the destination line number, the move is done before the first, or after the last, line having the specified line number. The current line becomes the first line moved.

Format: M[I][N][line1,[line2,]dline]

where:

I inserts the lines before, rather than after, the destination lines.

N does not list result.

dline destination line: when only the destination line number is used, the current line is moved to follow dline.

line1 when only two line numbers are used, line1 refers to the lines to be moved to follow dline.

line2 when three line numbers are used, line1 and line2 refer to a range of line numbers to be moved to follow dline.

Examples:

1. M64 = move the current line to follow lines numbered 64.
2. M32,64 = move lines numbered 32 to follow lines numbered 64.
3. MI32,38,64 = move line numbers 32 through 38 to the position before lines numbered 64. (The I option applies only to the destination.)

4.5.11 P (Line Feed): Provide Spacing

The P command issues a top-of-form or a given number of line feeds to the L0 device. This command is used to provide spacing between lines or pages printed by the Source Editor. The current line pointer is not changed.

Format: P[number of spaces]

where:

- P with no parameters, issues a top-of-form to the L0 device.
- P followed by a number, advances the L0 device by the given number of line feeds.

Examples:

- P = do a top of form on the L0.
- P12 = advance the L0 by 12 line feeds.

4.5.12 PLUS (+): Reposition Current Line

This command--which is a plus sign (+) and not a letter--repositions the current line pointer n lines past its present position.

Format: +[N][P][S][V][,number of lines]

where:

- N does not list results.
- P prints on the L0 device.
- S suppresses blanks on listing.
- V lists the vicinity of the current line.

number of lines indicates the number of lines past the current record that the user wants to go to.

Examples:

1. + = advance current line pointer to the next line.
2. +8 = go ahead 8 records.
3. +V16 = go ahead 16 records and list vicinity.

4.5.13 SUBSTITUTE: Replace One Character String with Another

The SUBSTITUTE command matches a given character string with text in the edit buffer and replaces the matched string with another character string. If no line number or range of line numbers is given, the current line is searched. If string1 or string2 contain a slash, another nonalphanumeric delimiter must be used in the command.

Format: S/string1/string2/[B][L][N][P][S][D][G][V][W][F][X] A
[,line1[,line2]]

where:

- | | |
|-------------------|--|
| /string1/string2/ | string1 is the string of characters (search string) to match with the text in the edit buffer. String2 is the string of characters (substitute string) which replaces the search string. |
| B | multiple blanks are treated as one blank in a search and in SUBSTITUTE. |
| L | is the delimited string option. See Y-DELIMIT command. |
| N | does not list result. |
| P | prints on the LO device. |
| S | suppresses listing multiple blanks. |
| D | after substitution, deletes remaining characters on the line. |
| A | all lines in the edit buffer (equivalent to ,B,E). |
| G | wildcard: @ matches any single character. |
| V | lists vicinity around line containing string1. |

W walks through each occurrence of the matched string in the specified range and gives the SXD prompt, which points to the occurrence and allows the user four options:

S = Substitute using given string

X = Stop SUBSTITUTE command immediately

D = Substitute, using the new string entered after dynamic substitute string prompt, DSS

Carriage Return = skip over occurrence.

F "First on a line." If specified, this option "finds" only the first instance in every line in the range. The option default, however, changes every instance of the string within the range.

X substitutes end of line, regardless of string1.

line1 used alone, refers to all records having that line number.

line2 used with line1, refers to a range of records from line 1 through line2.

If neither line1 nor line2 is specified, will search only the current line.

Examples:

1. S/123/ABC/ = search the current line. Substitute ABC for all occurrences of 123.
2. S/123/ABC/A = substitute ABC for 123 in all lines, all occurrences.
3. S/123/ABC/1,20 = substitute in lines numbered 1 through lines numbered 20 all occurrences of 123 with ABC.

Option Combinations:

S/123/ABC/AF = search all lines of the edit buffer: substitute ABC only for the first occurrence of 123 in each line in the edit buffer.

S/123/ABC/AW = step through all lines of the edit buffer, stopping at each occurrence of 123. The user is prompted by <SXD>:

S = Substitute using given string

X = Stop SUBSTITUTE command immediately

D = Substitute, using the new string entered after dynamic substitute string prompt, DSS.

After the carriage return, the Source Editor completes the substitution and displays the next occurrence.

S/123/ABC/D = substitute string 123 with string ABC on the current line and delete all characters remaining on that line.

S//123/1,20 = add string 123 to the beginning of lines 1 through 20.

S//123/X = appends the string 123 to the end of the current line.

4.5.14 UNBANK: Insert the Bank Buffer Text into the Edit Buffer

The UNBANK command inserts all lines in the bank buffer into the edit buffer. The Source Editor copies the data from the bank buffer to the location directly following the last line with the given line number, or following the current line if no line number is given.

A feature of the BANK/UNBANK command combination is that data can be banked during one read and unbanked after a NEXT (N) command refills the edit buffer with new text. The bank buffer is cleared following any terminate (T or ^) command. The first unbanked line becomes the current line.

Format 1: U[I][K][,line1]

Format 2: U[D]

where:

- I inserts the data from the bank buffer into a location directly in front of the specified target lines.
- K keeps the buffer intact during an unbank; otherwise, the buffer is cleared. Using the K option, a user can unbank the same buffer several times.
- D clears (deletes) the bank buffer without inserting the bank into the edit buffer. The D and K options are mutually exclusive.
- line1 when used, specifies all lines with the given number as target lines. If line1 is not specified, the target line is assumed to be the current line.

Examples:

1. UI = insert buffer immediately before the current line.
2. UK10 = unbank buffer, but keep it, and insert it directly after line number 10.
3. UD = clear the bank buffer without insertions.

4.5.15 Y-DELIMIT: Specify Delimiters for a Delimited String Search

The Y-DELIMIT command allows the user to specify characters to delimit search strings in the FIND and SUBSTITUTE commands. It is useful for doing token (identifier) matching. For example, adding a colon (:) to the buffer allows easier searches in programs written in assembly language.

If the L option is used in either of the two commands, a match between the search string and the record must satisfy the following conditions:

- a. Each alphanumeric character in the delimiter buffer is considered to be a valid delimiter.
- b. Each alphanumeric character not in the delimiter buffer is considered to be an invalid character delimiter.
- c. Each nonalphanumeric character in the delimiter buffer is considered to be an invalid delimiter.
- d. Each nonalphanumeric character not in the delimiter buffer is considered a valid delimiter (where $\emptyset$ = blank).

A string with a valid character on either side of it is considered a match. All Y-DELIMIT commands list the current contents of the delimiter buffer.

Format: Y/chars/[I,D]

where:

- | | |
|-------|--|
| chars | indicates the list of characters added or deleted from the delimiter buffer. |
| I | indicates that the specified characters are to be added to the delimiter buffer. |
| D | deletes the specified characters from the delimiter buffer. |

The Y command defaults to listing the contents of the delimiter buffer.

Examples:

1. Y = list current delimiters.
2. Y/ABC = clear delimiter buffer and insert ABC. Now the command F/XYZ/L can find XYZØ or ØXYZB as a match but not WXYZØ. (Ø is a valid char, since it is not in the buffer.)
3. Y/EF/I = keep contents of delimiter buffer and add E and F.
4. Y/ACE/D = remove characters A, C, and E from the delimiter buffer.
5. Y/*/I = add "*" to the delimiter's buffer. The command F/ABC/L would find =ABCØ as a match, but not *ABCØ.

4.5.16 Z: Go Back

The Z command causes the current line pointer to move back to some line preceding the current line. (If there are not enough lines in front of the current line, the current line pointer moves to the first line of the edit buffer.)

Format: Z[N][P][S][V][,number of lines]

An optional format uses a minus sign instead of the Z:

-[N][P][S][V][,number of lines]

where:

- N does not list results.
- P prints on the LO device.
- S suppresses listing of multiple blanks (list as one blank).
- V lists vicinity.

If the number of lines is specified, the Z command causes the current line pointer to back up that many lines. If number of lines is not specified, the current line pointer is set to the line preceding the current line.

Examples:

- Z = go back one line.
- 5 = go back five lines.
- Z10 = go back 10 lines.

4.6 EDIT BUFFER DATA TRANSFER COMMANDS

The following commands are used to copy data into the edit buffer and out to the S0 medium. The commands are listed in alphabetical order.

If it is necessary to read the next part of the file into the edit buffer, either the READ or the NEXT command can be used. The NEXT command first writes the contents of the edit buffer to the S0 and then reads the records from the SI, making the previous contents of the edit buffer inaccessible.

4.6.1 NEXT: Write Next Section of Standard Input File to Edit Buffer

The NEXT command writes the contents of the edit buffer to the S0 and refills the edit buffer with input from the SI, either to the specified record number, or to the input limit, or to the end of file. This command is normally used when an entire file is too large to fit into the edit buffer at one time. The first line of the edit buffer becomes the current line.

Format: N[G][N][,recordn]

where:

G "garbage": text read from SI is not put into the edit buffer, but is skipped.

N does not list results.

recordn refers to the ending record number specified by the user.

Examples:

N = write edit buffer to S0 and fill again to input limit or end of file.

N,600 = write edit buffer to S0 and fill again through record number 600.

NG = destroy the rest of the data in the SI, keeping only what has been written to the S0 and the current contents of the edit buffer.

N5
N4 = this input results in error. Record 5 has been read, so record 4 is inaccessible.

4.6.2 READ: Copy Source Text into the Edit Buffer

The READ command copies text from the input file (SI) to the edit buffer.

Each line of text read into the edit buffer is assigned a line number corresponding to its record number. For example, record number 53 becomes a line with line number 53. The first line read becomes the current line.

The G option causes entered text to be ignored and excluded from the new file. Otherwise, the text is stored consecutively in the edit buffer.

The recordn option allows the user to select the records to be input. The Source Editor reads into the edit buffer only the records from the last record read through the specified record number, the input limit, or the end of file, whichever comes first.

Format: R[G][N][recordn]

where:

- G "garbage": records (lines of text) are ignored and excluded from the edit buffer.
- N does not list results.
- recordn specifies the maximum record number desired to be read into the edit buffer; default is the last record of the file.

Examples:

1. R = copy SI to edit buffer until input limit or end of file.
2. R30 = read into the edit buffer the records following the last record read from the SI through record number 30.
3. RG30 = ignore current record in SI through record 30. (The next read will begin at record number 31.)

4.6.3 WRITE: Write Text to the Output File (S0)

The WRITE command removes specified lines from the edit buffer and writes them to the S0. (If used, the K option keeps the specified records in the edit buffer.) When large files are edited, the WRITE command can make space available in the edit buffer for additional records to be input.

On termination, the Source Editor appends the contents of the edit buffer to data previously written to the S0. Thus, a WRITE is not required at that time. The current line becomes the line following the last line written.

Format: W[K][N][,line1[,line2]]

where:

- K keeps a copy of lines in the edit buffer.
- N does not list results.
- line1 used alone, refers to all lines having that line number.
- line2 used with line1, refers to a range of lines from line1 through line2.

If neither line1 nor line2 is specified, the command writes the current line.

Examples:

- WK = write the current line to the S0, but keep a copy of the line
 in the edit buffer also.
- W62 = write all lines having line number 62 to the S0.
- WN1,62 = write lines number 1 through 62 to the S0, removing them from
 the edit buffer; do not list the lines.

4.7 EDIT TERMINATION COMMANDS

When a user enters the TERMINATE (T) command, the following occurs:

1. The edit buffer is appended to output.
2. The remaining source file (not yet read) is appended to the output.
3. The files (new and original) are renamed, if necessary.

When edit was initiated, the Source Editor created a scratch file for standard output. When the edit is terminated successfully, one of two possibilities in file naming exist:

1. Automatic backup--where the original source file is renamed to have the file extension .BAK, and the scratch file is named as the original file. (E.g., user entered EDIT file1.)

or

2. The new file created from the source file keeps the original name, and the scratch file is renamed with the new filename. (E.g., user entered EDIT file1=file2 or EDIT file1=TV.)

For an abnormal termination, the scratch file is deleted and the original source file is not affected.

4.7.1 TERMINATE: End the Editing Process

The TERMINATE command is the final command entered during an editing process. This command completes the creation of the output file and terminates the Source Editor. At termination, the Source Editor writes the edit buffer to the S0 and then appends the remaining source text in the SI device to the S0 device. This file is created with the same base name as the source file, but with an extension of .EDT. The Source Editor searches the SI device for a file of the same base name and an extension of .BAK. If one is found it is deleted. The Source Editor then renames the original source file with the extension .BAK. At this time, the newly created .EDT file on the S0 is renamed with the original extension of the source file. Renaming variations occur when different formats of the EDIT commands are invoked. When the renaming process is complete, the Source Editor returns control to the OS4 interface.

If T is entered when no changes have been made to a file, the user is prompted as follows:

NO CHANGE--DO YOU WANT TO TERMINATE? IF YES, TYPE T.

If the user responds with T, normal termination follows.

Format: T

Examples:

1. T = terminate the edit process.
2. TG = keep only contents of the S0 and the edit buffer and ignore remaining text in the SI.

4.7.2 Aborting the Edit Process

As a safety feature, a terminate (T) request following a "D,B,E" command, when nothing has been written to S0 and nothing is left to be read from SI, is treated as an ABORT request.

Format 1: T [G][X]

where:

- G causes only the contents of the edit buffer to be appended to the text already written to the S0. Any remaining source text in the SI is ignored.
- X causes the termination to abort as if a ^ had been entered if no changes were made to the source, but terminates normally if changes have been made.

The Source Editor also has a command to abort the edit, leaving the source text in its original state. The command is the ABORT command, whose symbol is the caret (^) or K.

Format 2: ^

Format 3: K

If ^ or K is typed after changes have been made to a file, the user is prompted as follows:

CHANGES MADE--DO YOU WANT TO ABORT? IF YES, TYPE K.

If the user responds with K, the Source Editor does an abnormal exit leaving the original file intact.

Examples:

1. ^ = abort the edit process and leave the source file in its original state.
2. K = response to a ^ when changes have been made.

APPENDIX A
MESSAGE SUMMARY

The errors, warnings, and other messages contained in this summary are displayed on the operator terminal when the specified conditions are detected.

ERRORS

*** BUFFER CAPACITY ERROR ***

Input to the bank buffer exceeds its limit.

*** DEVICE ERROR ***

Hardware error on one of the assigned devices.

ERROR IN RENAME--CHECK FILES

Generated if OS4 tries to rename a file with a filename that already exists.

*** LINE # ERROR ***

User has specified an illegal line number.

*** SYNTAX ERROR ***

An error in the format of the command has been detected.

?????

The source text file listed in the EDIT call cannot be located on the given device.

WARNINGS

EDIT BUFFER EMPTY

All text has been removed from the edit buffer and an attempt has been made to execute a command on data in the edit buffer.

*** INVALID IN CREATE MODE ***

Create mode is in effect and the user has issued a command which is valid only in edit mode.

*** INVALID IN EDIT MODE ***

Edit mode is in effect and the user has issued a command which is valid only in create mode.

MESSAGES

*** BANK BUFFER EMPTY ***

UNBANK command has been attempted when the bank buffer is empty.

BYTES AVAILABLE

Displayed when the user issues the E command asking for the available memory in bytes.

CURRENT GLOBAL OPTIONS

Displayed when the O command is issued, followed by the modes that are in effect.

*** EDIT BUFFER FULL ***

Generated when inserted lines fill the edit buffer to its maximum capacity.

EDT...

Indicates that operations specified in the previous EDIT command line are complete and that the Source Editor is ready to accept additional input.

END OF DATA

Indicates that the edit buffer pointer is currently past the last line of buffered data.

*** END OF FILE ***

Displayed when the current line is the last line in the edit buffer and an attempt is made to go beyond it.

END OF FILE REACHED

End of file has been reached before the requested line number in a READ command.

INPUT BUFFER LIMIT REACHED

Edit buffer is 75 percent full before end of file or the requested line number is reached.

THESE ARE NOT VALID DELIMITERS

Displayed when the Y command is issued. Gives the characters that are not allowed as delimiters for a matched character string.

REQUESTED LINE # REACHED

Displayed on a READ command when text from the current line to the specified line has been copied to the edit buffer.

THESE ARE VALID ALPHANUMERIC DELIMITERS

When the Y command is given, displays the characters currently acceptable as delimiters for a matched character string.

LINE TRUNCATED TO 132 CHARACTERS

Displayed when more than 132 characters are input in insert or input mode with a carriage return <CR>.

LINE OVERFLOW *** ABORTED ***

Displayed during a substitution. When the resulting string is greater than 132 characters, the line is left unchanged.

NO CHANGE--DO YOU WANT TO TERMINATE? IF YES TYPE T

Displayed if the user has attempted to terminate from a file that has not been changed. This is a warning that the user is about to destroy the .BAK file without updating the original.

CHANGES MADE--DO YOU WANT TO ABORT? IF YES TYPE K

Displayed when the user has attempted to abort from a file after changes have been made. It is a warning that the user may be about ready to destroy all that he has edited since his last termination.

ABNORMAL EXIT

Displayed when the user has just aborted from the editor following a TX or ^ command.

APPENDIX B

SYNTAX SUMMARIES

B.0 EDIT COMMANDS

The following are alphabetical summaries of the edit commands and options.

<u>Command</u>	<u>Format</u>
Align	A
Bank	B[K][N][,line1[,line2]]
Copy	C[I][N][[line1[,line2,]]dline]
Delete	D[N][P][R][,line1[,line2]]
Display Memory	E
Find	F/string1/[W][B][G][L][N][P][S][V] ^A [,line1[,line2]]
Go Back	Z[N][P][S][V][,number of lines] -[N][P][S][V][,number of lines]
Go Forward	+ [N][P][S][V][,number of lines] Carriage Return
Go To	G[N][P][S][V][,line1]
Insert	I[I][line1]
List	L ^V _W [P][S] ^A [,line1[,line2]]
List Global Options	O
Move	M[I][N][[line1[,line2,]]dline]

<u>Command</u>	<u>Format</u>
Next	N[G][N][,recordn]
Provide Spacing	P[number of spaces]
Quit Options	Q[S][B][C][P][N]][L][V][W]
Read	R[G][N][recordn]
Set Options	O[S][B][C][P][N]][L][V][W]
Specify Delimiters	Y/chars/[I,D]
Substitute	S/string1/string2/B][L][N][P][S][D][G][V][W][F] [X] ^A [,line1[,line2]]
Terminate	T[G][X]
Terminate (abort)	^ K
Unbank	U[I][K][,line1] U[D]
Write	W[K][N][,line1[,line2]]

B.1 COMMAND LINE OPTIONS

<u>Option</u>	<u>Meaning</u>
A	All occurrences within range of lines
B	Multiple blanks seen as single blanks
C	Create
D	Destroy
	Delete delimiters
	Delete to end of line
F	First occurrence in line
G	Garbage
	Wildcard
I	Immediate insert
	Insert delimiters
K	Keep in edit or bank buffer (else purge)
L	Delimited string
N	No list
P	Print on LO device
R	Replace
S	Suppress expansion of multiple blanks
V	Vicinity (± 11 lines)
W	Walkthru
X	Extend

