

SOURCE EDITOR REFERENCE MANUAL

93465-90

SEPTEMBER 1980



Engineering Reference Number
90-93465-00F0

NAKED MINI® Division

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

Telephone Rickmansworth 71211

Telex 92265

© 1980 Computer Automation, Inc.
Printed in U.S.A.

REVISION HISTORY

<u>Revision</u>	<u>Issue Date</u>	<u>Comments</u>
F0	Sept. 1980	Original issue of standalone manual. Released at F0 to correspond with revision level F1 of software.

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. The Source Editor operates under the OS4 Operating System and enables a user to create or revise text in files.

This manual can serve both as a learning tool for programmers new to the Source Editor and as a reference source for experienced users. Section 1 introduces the Source Editor and its use. Functional operations of the Source Editor are described in Section 2. Section 3 describes the five methods of invoking the Source Editor, the general use of the Source Editor commands, and a method for editing large files.

Descriptions of the Source Editor commands and examples of their use are provided in Section 4.

Appendixes to the manual provide a list of errors, warnings and other messages, as well as a format comparison between the E0 revision and the F0 revision of the Source Editor.

The OS4 System User's Manual (93460-90) contains information related to the use of the Source Editor Utility. Appendix B of the OS4 System User's Manual is replaced by the release of this manual.

Contact your Computer Automation, Inc. representative for copies of these documents.

CONTENTS

SECTION 1	INTRODUCTION: SOURCE EDITOR UTILITY	1-1
	1.1 OPERATIONS OVERVIEW	1-1
	1.2 OPERATING ENVIRONMENT	1-1
	1.3 DOCUMENTATION CONVENTIONS	1-2
SECTION 2	FUNCTIONAL OVERVIEW.	2-1
	2.1 INPUT TO THE EDIT BUFFER.	2-2
	2.2 THE 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 TERMINATING THE EDIT.	2-5
	2.5 MODES OF OPERATION.	2-5
SECTION 3	COMMUNICATION WITH THE SOURCE EDITOR	3-1
	3.1 INVOKING THE SOURCE EDITOR.	3-1
	3.1.1 Logical Unit Assignments	3-1
	3.1.2 Editing a Previously-Created File	3-2
	3.1.3 Editing One File With the Contents of a Second File	3-4
	3.1.4 Creating a New File From Terminal Input.	3-5
	3.1.5 Creating a New File From One or More Existing Files.	3-5
	3.1.6 Editing Several Files at One Time.	3-6
	3.1.7 Editing Large Files.	3-7
SECTION 4	THE EDIT COMMANDS.	4-1
	4.1 COMMAND CONSTRUCTION.	4-1
	4.2 THE OPTIONS	4-2
	4.3 INITIATING AN EDIT.	4-3
	4.4 THE EDIT BUFFER COMMANDS.	4-4
	4.4.1 ALIGN: Renumber the Edit Buffer (Create Mode)	4-4
	4.4.2 BANK: Save Records To Be Inserted Elsewhere	4-4
	4.4.3 DELETE: Remove Records From the Edit Buffer	4-5
	4.4.4 E: Display Available Memory in Bytes.	4-6
	4.4.5 FIND: Locate a User Specified String in the Edit Buffer	4-6

4.4.6	GO TO: Reposition Current Line	4-8
4.4.7	INSERT: Insert Records	4-8
4.4.8	LIST: List Records to the SL	4-9
4.4.9	Mode: Set Global Options	4-10
4.4.10	MOVE: Move Records from Location to Location.	4-13
4.4.11	P (Line Feed): Provide Spacing	4-13
4.4.12	Q: Clear Global Options.	4-14
4.4.13	SUBSTITUTE: Replace One Character String With Another	4-15
4.4.14	UNBANK: Insert the Bank Buffer Records . . .	4-16
4.4.15	Y-DELIMIT: Specify Delimiters for a Delimited String Search	4-17
4.4.16	Z: Backup One Record	4-18
4.5	THE EDIT BUFFER DATA TRANSFER COMMANDS	4-19
4.5.1	READ: Copy Source Text Into the Edit Buffer	4-19
4.5.2	WRITE: Write Records to the Output File (SO).	4-20
4.5.3	NEXT: NEXT Section of SI Written to Edit Buffer.	4-21
4.6	THE EDIT TERMINATION COMMANDS	4-21
4.6.1	TERMINATE: Ending the Editing Process. . . .	4-21

APPENDIXES

A	MESSAGE SUMMARY.	A-1
B	FORMAT COMPARISON BETWEEN E100 REVISION AND F100 OF THE SOURCE EDITOR.	B-1

FIGURES

<u>Figure</u>		<u>Page</u>
2-1	Source Editor Functional Overview.	2-1
2-2	Banking Data	2-3
2-3	Output From the Edit Buffer.	2-4

SECTION 1

INTRODUCTION: SOURCE EDITOR UTILITY

The Source Editor utility is a line oriented text editor which allows the user to create new text or revise existing text from a file. The Source Editor allows the user either to enter commands interactively from a terminal or to read edit commands from a file.

1.1 OPERATIONS OVERVIEW

The Source Editor provides commands for reading source text into an internal 'edit buffer', operating on those lines in the buffer, and writing the results of the edit to an output file. The utility supports an internal bank for retaining lines along with commands which transfer text between the bank and the edit buffer. Other operations include:

- Inserting, deleting, changing, and moving lines in the edit buffer.
- Character string searching and substituting with range control and delimiter control.
- Listing and printing 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.

1.2 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@ 4 OS4 Operating System
(D0 revision level or later) Diskette ID No. F46001; CAI Model No. 93460-01

1.3 DOCUMENTATION CONVENTIONS

This manual uses the following conventions in presenting command formats:

- A Backspace character deletes the previous character; so "AA<backspace>CD" is the same as "ACD". A carriage return after a backspace deletes the entire line.
- A syntax element written in upper case 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 entity can be repeated one or more times. Ellipses are not part of the input.
- Lower-case 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:

$$\begin{Bmatrix} \text{pun} \\ \text{lun} \end{Bmatrix}$$

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 in any of the forms described in the preceding subsection. That is, the term "file" is used in place of the construction:

$$\begin{Bmatrix} \text{pun.} \\ \text{lun.} \end{Bmatrix} \text{name[.extension]}$$

to indicate that the user enters a file name 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, .~~XXX~~. 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:

$$\begin{Bmatrix} \text{device} \\ \text{file} \end{Bmatrix}$$

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 OVERVIEW

This Source Editor overview explains the use of the edit buffer, the assignment of line numbers, and the concept of the current line. This section discusses edit operations and describes the six modes of operations available to the user. Each mode modifies the manner in which the commands operate on the edit buffer.

Edit operations are completed in four phases (Figure 2-1).

1. Copying source text from the input medium (SI) to the edit buffer.
2. Performing edits on the lines of source text in the edit buffer.
3. Copying the edited text from the edit buffer to the output medium (SO).
4. Terminating the edit.

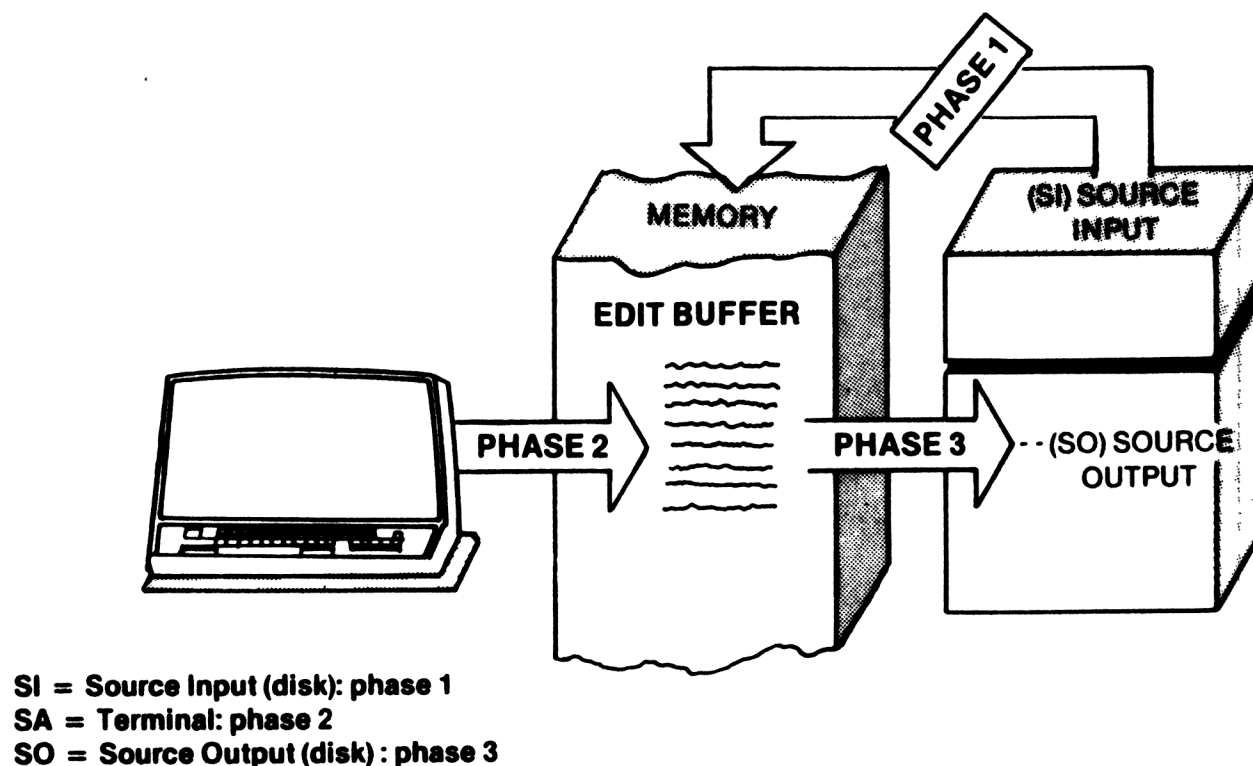


Figure 2-1. Source Editor Functional Overview

The Source Editor is initiated by a command and all the 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 THE EDIT BUFFER

A source file has one record per line. The user enters the (R) 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 S0 file. If there are more records in the source file, the next read starts the transfer with the record following the last record read.

An "RG" command 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 re-read records in the source file during an edit process.

2.2 THE 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 source output (S0) and the appropriate termination command is issued.

The same edit buffer is used when creating new text. The Source Editor generates an input prompt which signals the user to enter data. The user can end input at any time and 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 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 as opposed to sequential records in the Source Input (SI) and Source Output (S0) files. In the edit buffer, two or more lines can have the same line number meaning that a line number is not identical to a line.

No new line numbers are created by editing; however, line numbers can be deleted. For instance, when the source file is read into the buffer, all records and line numbers correspond accordingly. 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. So for insert, the Source Editor assigns the inserted line the same number as the line following the insert or the same number as the line prior to the insert (the specific line number assignment depends on the command line input). When lines are written to the source output, they are ordered as sequential records according to the order in which they were written. It is only when the file is read again to the edit buffer that the records are assigned new line numbers ascending in numerical order.

There is a special mode that allows lines in the edit buffer to be sequentially renumbered. This mode (create) is useful primarily for creating large segments of new text within an old file, where it is awkward for the new lines to all 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 input is completed, the Source Editor commands can be used to edit the text as if the text 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.

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 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 list of the edit buffer, the current line is indicated by an equal 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. The bank buffer communicates solely with the edit buffer so text cannot be moved to the bank buffer and then written to the Source Output medium. 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 must be unbanked to the edit buffer at the same time. (Figure 2-2).

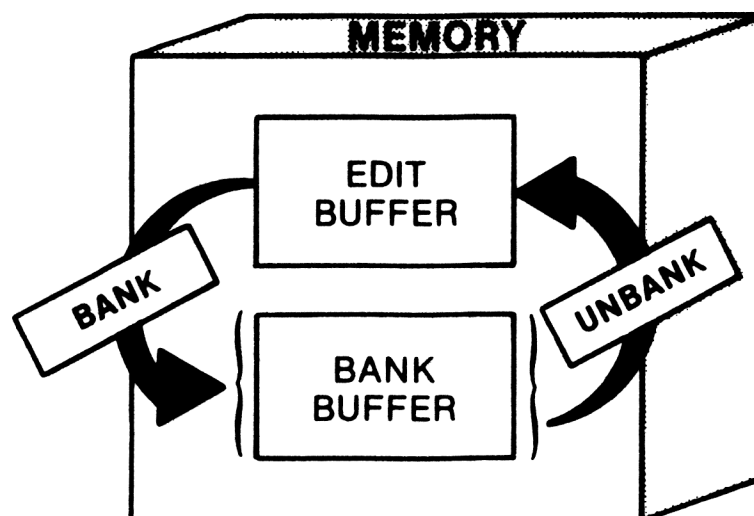


Figure 2-2. Banking Data

2.3 OUTPUT FROM THE EDIT BUFFER

When a user issues a WRITE command, the specified line or lines are written to the standard output device. Each line written becomes one record. The 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. If the file A is again READ into the edit buffer, the data appears as shown.

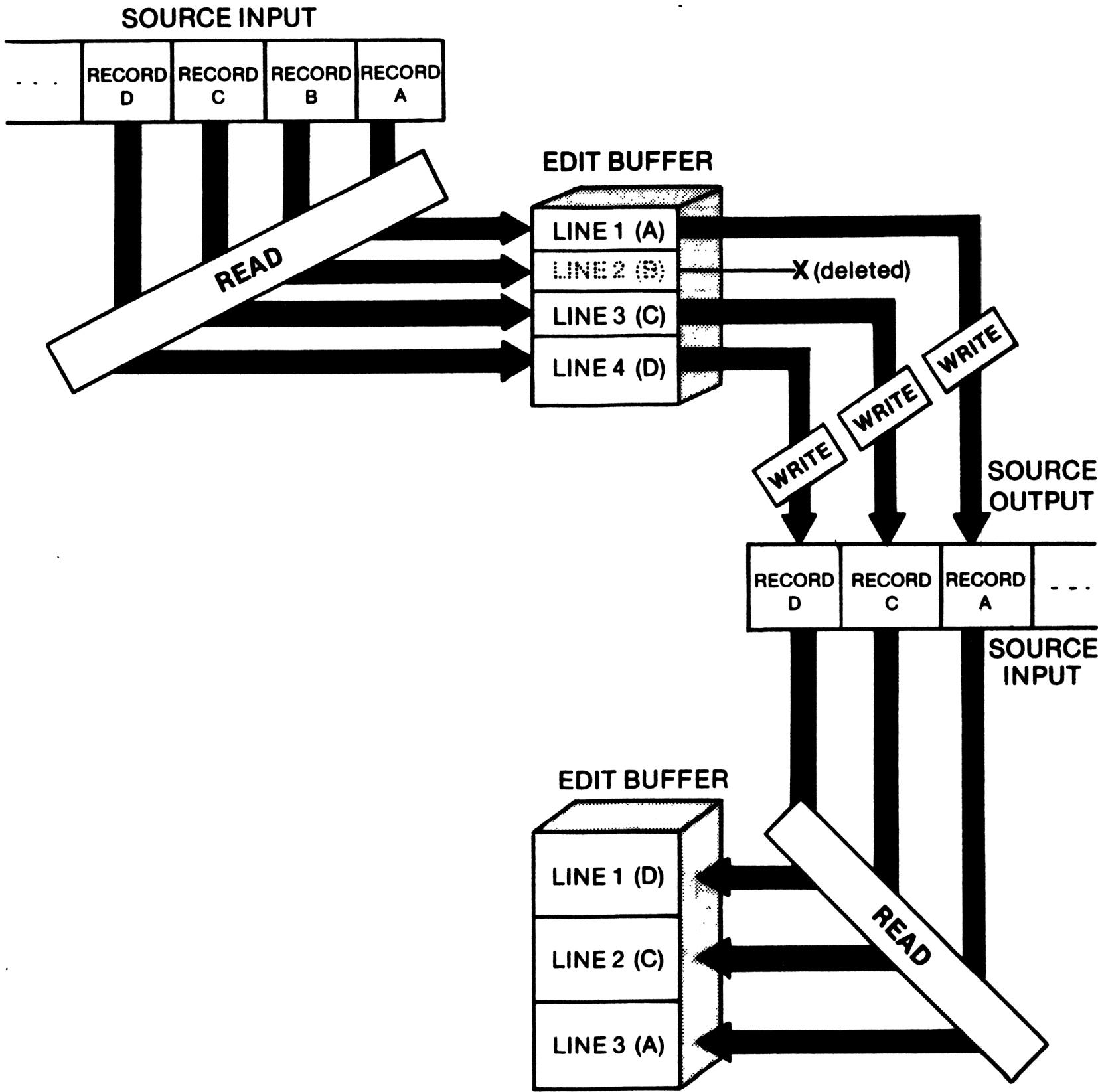


Figure 2-3. Output From the Edit Buffer

2.4 TERMINATING THE EDIT

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

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

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

- A. The automatic back-up where the source file is renamed to have the file extension .BAK, and the scratch file is named as the original file.
- B. The new file where the source file keeps the original name, and the scratch file is renamed with the new file name.

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

2.5 MODES OF OPERATION

The Source Editor supports six global options or modes. When a particular mode is set, all commands with that option operate as if the option were entered for each command. One to six modes can be set, reset, or terminated at any one time.

- The Edit/Create Mode defines whether new text is being created or existing text is being edited.
- The Delimiter Mode determines whether character strings are searched for with or without delimiters.
- The Blank Mode modifies the way multiple blanks in the text are treated by the commands.
- The Suppress Mode determines how multiple blanks are listed on the terminal and the printer.
- The Print Mode controls whether or not the results of the commands are printed on the LO device.
- The No List Mode defines whether or not results of the commands are listed on the terminal.

SECTION 3

COMMUNICATION WITH THE SOURCE EDITOR

The Source Editor runs under the OS4 Operating System. Further information about the OS4 Operating System can be found in the OS4 System User's Manual, CAI Model Number 93460-90. This section describes the six methods of invoking the Source Editor and examples of EDIT use. The six methods are as follows:

- Editing a previously-created file.
- Editing one file with the contents of a second file.
- Creating a new file with input from the terminal.
- Creating a new file from one or more previously-created files.
- Editing several files at one time (the general form of the EDIT).
- Editing large files.

3.1 INVOKING THE SOURCE EDITOR

The Source Editor is invoked by an OS4 command, described in this subsection. Before the command is issued, the user should be sure that the logical units used by the Source Editor are assigned to appropriate physical units.

3.1.1 Logical Unit Assignments

The following Logical Units are used by the Source Editor for its operations.

- SF The System File unit supplies the Source Editor loadable binary code.
- SI The Standard Input unit supplies the source text.
- LO The List Output unit receives listings generated by the "Print" command.
- SL The System Log displays output from the Source Editor commands.
- SO The Standard Output unit receives the modified version of text from the edit buffer.
- SA The Standard Alternate Unit is the input device which supplies the edit commands to the Source Editor.
- CI The Command Input supplies the EDIT command line to OS4.
- UF The User File provides a default location for the source text which is searched when neither a pun or a lun is given in the file name.

3.1.2 Editing a Previously-Created File

The Source Editor provides two formats to invoke 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 unit (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.

Format 1: Using Default Assignments

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

Format: $\left[\begin{matrix} \text{pun} \\ \text{lun} \end{matrix} \right].$ EDIT file

where:

pun is the parameter for the physical unit on which the Source Editor is located. If neither a pun or lun is specified, the SF unit is searched for the Source Editor.

lun is the parameter for the logical unit on which the Source Editor resides. If neither a pun or lun is specified, the Source Editor is read from the SF unit.

file is the parameter for the file to be edited. An OS4 file identifier is comprised of an optional physical or logical unit name, followed by a period, a mandatory file name of up to 11 alphanumeric characters, and an optional period and three character extension. If neither a physical or logical unit name is included in a file identifier, the UF is searched for the named file. 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, .~~XXX~~. If none of these versions is found, an error message is displayed.

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.

Format 2: User Provided Assignments

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

Format: $\left[\begin{Bmatrix} \text{pun} \\ \text{lun} \end{Bmatrix} \right] \text{ EDIT}$

EXAMPLE:

Assume that the programmer has issued the following ASSIGN commands (Section 4, OS4 System User's Manual) 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 Editor utility program from the SF unit, prompts for editing commands on the 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.

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, and the user then can perform any of the edit functions on this file. When the terminate command is entered, the edit buffer is written to the SO unit and labeled with the first file name given in the EDIT invocation. The Source Editor recognizes the comma following the first file name 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 entered in the EDIT invocation and delimited by commas are edited separately from the others. Each receives its original file name when written to the SO unit. The file name and extension must be entered if the extension is not listed in Section 1.3. If the file name is not found, the EDIT command aborts and control returns to OS4.

This function of the Source Editor is invoked by a command of the form

Format: $\left[\begin{Bmatrix} \text{pun} \\ \text{lun} \end{Bmatrix} \right] \text{ EDIT file1,file2, ...}$

The user inputs all files to be edited during the EDIT invocation, 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.

See Format 1 of Section 3.1.2 for a description of the command parameters: lun, pun, file.

EXAMPLE:

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

3.1.3 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.

This function of the Source Editor is invoked by a command of the form

Format: $\left[\begin{Bmatrix} \text{pun} \\ \text{lun} \end{Bmatrix} \right] \text{ EDIT medium1}(\text{medium2})$

where:

medium1 is the source text to be edited.

medium2 contains the edit commands used to edit Medium1.

See Section 3.1.2 for a description of the other command parameters: lun, pun, file.

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.

3.1.4 Creating a New File From Terminal Input

The user can create a new file from terminal input while using the Source Editor. First, the user inputs the EDIT command as below, and 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 the <INP> prompt. 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 file name and extension given in the Source Editor command.

This function of the Source Editor is invoked by a command of the form

Format: $\left[\begin{matrix} \text{pun} \\ \text{lun} \end{matrix} \right] \cdot$ 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.

3.1.5 Creating a New File From One or More Existing Files

By involving the Source Editor, the user can 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 it the file name specified by the user. The Source Editor then checks for a plus sign; 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. 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 present file in the edit buffer. To work with the entire merged file, the user must terminate and invoke the Source Editor with the new file name.

These functions are invoked by a command of the form:

Format: $\left[\begin{matrix} \text{pun} \\ \text{lun} \end{matrix} \right] \cdot$ 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

file2,3 are the names of the original files to be appended to file1.

The other parameters, pun and lun, are described in Section 3.1.2.

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, resulting in one file called DF1.MERGE.ASM.

3.1.6 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.

Format: EDIT [parameter[,parameter]...]

where:

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

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

EXAMPLE 1:

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

This command causes the Source Editor to prompt for editing commands on the Operator Console (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.

EXAMPLE 2:

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

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

EXAMPLE 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 file name. Then, the Source Editor edits DF.PROG with input from the operator console (OC) and writes the edited text to DF1 with the file name of DF1.PROG.

3.1.7 Editing Large Files

The Source Editor can handle 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. For example, the edit buffer of a 128K Byte machine can hold approximately 1800 lines of assembly language or 2300 lines of COBOL. 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 can enter the NEXT command. This command writes the edit buffer to the 'S0' and copies the next portion of the source text from the 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. (One command, however, will cross the bounds of the two portions: Unbank. This command allows the user to move a set of data in 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

THE EDIT COMMANDS

The Source Editor offers a broad range of commands to modify text and to aid the user. This section describes the command construction, including arguments, options and delimiters. The option set is standard to all commands and is listed in this section. Following the option set, this section provides the EDIT command descriptions. These descriptions provide a general overview of the command and its format along with a few examples describing the uses of the command. Some commands allow particular options to be selected. An explanation of these options completes the description of each command. The commands are organized into three sections:

1. The Edit Buffer commands
2. Edit Buffer Data Transfer commands
3. Edit Termination commands

Appendix A lists the error messages that the user encounters and describes the condition that causes each message.

4.1 COMMAND CONSTRUCTION

A Source Editor command is composed of four fields. The first field is a one character symbol for the command. The second field is a string (used only in the FIND, SUBSTITUTE, or Y-DELIMIT commands). The third field consists of a list of options which expand the use of the command and the fourth field indicates the range of lines that the command affects. These fields can be separated by commas, if necessary; the command ends with a carriage return. Only one command per line is allowed.

The first character of a command is always the one letter symbol for the command, such as "I" for INSERT.

Slashes delimit the character string operands for the FIND, SUBSTITUTE and Y-DELIMIT commands. A slash indicates the beginning of the first string. A second slash separates the first string from the second string, if given. A third slash ends the string. This last slash can be omitted if no other characters follow.

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

Options for the commands, if chosen, further describe the original command, such as "F" for first one only. Each option is one character in length and can be selected in any order. No commas are allowed between the options.

The line number or range of line numbers is the last field to be entered in a command. If no range is given, the current line is used (except in the FIND command). A comma is required between two line numbers indicating a range. The command ends with a carriage return.

4.2 THE OPTIONS

The Source Editor options enhance and expand the use of the commands. The options are not standard to all commands, but they operate in a similar manner for each of the commands offering the particular options. Some of the options listed below are global options. They can be set globally by the Mode command or locally in each command. When these options are used locally, they temporarily override the default values of the global options. In the following list, (G) indicates that the option is both a local and a global option. The command options are:

- A Means 'all occurrences' within the range of lines (used as an option only in the FIND command).
- B Multiple blanks seen as single blanks (G).
- C Create Mode (G).
- D
 - (a) Destroy option for UNBANK.
 - (b) Delete delimiters option for Y-DELIMIT.
 - (e) Delete to end of line in SUBSTITUTE.
- F First occurrence in the line only (used in the SUBSTITUTE command).
- G Garbage, records are destroyed.
- I
 - (a) Immediate insert option for INSERT, MOVE, and UNBANK.
 - (b) Insert delimiters option for Y-DELIMIT.
- K
 - (a) Keep in edit buffer (else purge) for BANK and WRITE.
 - (b) Keep in bank buffer (else purge) for UNBANK.
- L Delimited string option (G).
- N No listing of result (G).
- P Print on the 'LO' (G).
- R Replace option for DELETE.
- S Suppress expansion of multiple blanks (G).
- V Vicinity (11 previous lines to 11 following lines) option for LIST.
- W Walk through, used in the FIND, SUBSTITUTE, and LIST commands.

NOTE

The symbols A, B, C, and E have different meanings when used in the range field of the command, rather than 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/ ~~B~~ A/A/B,23	SUBSTITUTE (using the blank option) in line 23.
S/ ~~B~~ A/A,B,23	SUBSTITUTE in lines from the beginning through line 23.

4.3 INITIATING AN EDIT

The Source Editor is loaded by the EDIT command line to OS4 (see Section 3). After loading, a program announcement appears on the CRT followed by the message "outputfile=inputfile". The outputfile refers to the destination medium of the edited source text and the 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 file name upon normal termination (T).

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

In most cases, the first command that is entered in EDIT is the READ command. The READ command 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 non-existent lines is prompted with an "end-of-data" message. If the TERMINATE command is entered before a READ command or an INSERT command, the EDIT is terminated with the resulting file unchanged.

After the READ command is 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 a slash, asterisk, "/*," is entered in response to the <INP> prompt. Otherwise, the Source Editor functions as explained above.

4.4 THE 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, etc. The commands are listed in alphabetical order.

4.4.1 ALIGN: Renumber the Edit Buffer (Create Mode)

The ALIGN command can be used only in Create Mode. 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 new lines with the same number.

Format: A

EXAMPLE:

A = renumbers the edit buffer

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.

4.4.2 BANK: Save Records To Be Inserted Elsewhere

The user can save, or bank, a line or a group of contiguous lines of text from the edit buffer. (A separate 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 the bank buffer is not retained after a TERMINATE (T) command is given. The current line becomes the line number of the line following the last line moved or copied.

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

where:

K "KEEP": banks specified lines while 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.

EXAMPLE:

B = bank the current line do not keep (move line to the edit buffer).

BK6 = bank lines with line number 6 and also keep in original location (copy lines to the edit buffer).

B6,12 = bank lines from line 6 to line 12; do not keep (move lines to the edit buffer).

4.4.3 DELETE: Remove Records 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. 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][R][,line1[,line2]]

where:

N suppresses a listing of results (local and global option).

R replace option for DELETE.

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.

EXAMPLE:

D = delete current line, no listing.

D140 = delete all lines with line number 140.

DR140,150 = delete all lines with line number 140 through line number 150, replacing them with user input.

4.4.4 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

EXAMPLE:

E = displays available memory in bytes.

4.4.5 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.

Format: F/String/ $\begin{bmatrix} A \\ W \end{bmatrix}$ [B][L][N][P][,line1[,line2]]

where:

/string/ is the character string desired. The string is enclosed in slashes (if the string contains a slash, then the string can be enclosed in a different alphanumeric character). The last slash is optional if no other parameters are on the line.

$\begin{bmatrix} A \\ W \end{bmatrix}$

This option provides 3 ways for range search:

1. A finds all occurrences of the string in the range, and displays them. This option does not change the current line pointer.

2. W provides a 'step through'. 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 display together with the prompt (xxx). If no more lines are found, the current line pointer is not affected by the command.
 3. For the default, the 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.
- B multiple blanks seen as single blanks (local and global option).
- L allows the user to specify that preset alphanumeric characters must bind the matching string of characters. The user also can specify that preset non-alphanumeric characters must not bind the matching string. The characters are preset with the Y-DELIMIT command.
- N no listing of result (local and global option).
- P print on the 'L0' device (local and global 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.

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

EXAMPLE:

- F/123 = find the first occurrence of the character string '123' in the entire edit buffer.
- 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.
- 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.

4.4.6 GO TO: Reposition Current Line

The command "GO TO" 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. (To retreat (backup) one line from the current line, the user enters the Z command.) If there is more than one line with the given line number, the pointer is positioned to the first one.

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

where:

N no listing of results (local and global option).

P print on the 'LO' device (local and global option).

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).

EXAMPLE:

G = advance current line pointer to the next line.

GN8 = reposition current line to line 8 with no listing.

Carriage
return = advance current line pointer to the next line.

4.4.7 INSERT: Insert Records

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 the command is given, the Source Editor displays an <INS> prompt indicating that the user can input data. A carriage return ends that line of input and displays a new <INS> prompt for continued entry. Input is terminated by 2 consecutive carriage returns or by a slash, asterisk, "/*". If more than one line share the given line number, the insertion occurs after the last line of the given line number.

The I option 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.

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.

A question mark line, "?", 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 insert 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.

EXAMPLE:

I = insert lines after the current record.

II,10 = insert lines immediately before line 10.

4.4.8 LIST: List Records to the SL

The LIST command lists specified lines of the edit buffer to the System Log (SL), 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] $\begin{bmatrix} A \\ \text{line1[,line2]} \end{bmatrix}$

V causes a listing of 11 lines prior to the current line, the current line, and 11 lines beyond the current line.

W 'steps through' the range of listed lines. A prompt <FXR> gives the user the option to:

F = display the next page (forward)
R = display the previous page (reverse)
X = terminate listing

- P print on the 'L0' device (local and global).
- S suppresses multiple blanks in a listing. (All multiple blanks printed as one blank.)
- A specifies listing 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.

EXAMPLE:

L = list the current line.

LP150 = list all lines numbered 150 and print them on the L0.

LS150,180 = list line number 150 through 180, suppressing multiple blanks (print only one blank for multiple blanks).

Option Combinations:

LV = list 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 butter through lines numbered 50.

4.4.9 Mode: Set Global Options

Six modes of operation were discussed in Section 2.4. When a global option is set, all commands with the option default to the option. Any of the six options can be set (or reset) at one time. The six conditions are explained below.

Format: O[S][B][C][P][N][L]

MODE

EXAMPLE:

0 = list current global options only.

S = Suppress

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

EXAMPLE:

0S = list current global options and sets suppress mode.

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.

EXAMPLE:

0B = list current global options and set blank mode.

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

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.

EXAMPLE:

0C = list current global options and set create mode

P = Print

When the print global option is set, the results of execution of commands with that option are printed on the L0 device.

EXAMPLE:

OP = list current global options and set print mode

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.

EXAMPLE:

ON = list current global options and set no list mode

L = Limiters

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's 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.)

EXAMPLE:

OL = list current global options and set limiters mode.

4.4.10 MOVE: Move Records From Location to Location

The MOVE command performs the Bank and Unbank operations in one command: the specified records are removed from their 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][line1,[line2,]]dline

where:

- I inserts the lines before, rather than after, the destination lines.
- dline destination line: when only the destination line number is used, the current line is moved to follow dline. "dline" must always be specified.
- 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.

EXAMPLE:

M64 = move the current line to follow lines numbered 64.

M32,64 = move lines numbered 32 to follow lines numbered 64.

MI32,38,64 = move line numbers 32 through 38 to the position before lines numbered 64. (The I option applies only to the destination.)

4.4.11 P (Line Feed): Provide Spacing

The P command issues a top-of-form or a given number of line feeds to the LO 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[no. 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.

EXAMPLE:

P = do a top of form on the L0.

P12 = advance the L0 by 12 line feeds.

4.4.12 Q: Clear Global Options

Six options are set globally and remain set until the Q command is used to clear the option.

Format: Q[B][C][L][N][P][S]

where:

- B multiple blanks seen as single blanks on search (local and global option).
- C create mode (local and global option).
- L delimited string option on search (local and global option).
- N no listing of result (local and global option).
- P print on the 'L0' device (local and global option).
- S multiple blanks listed as one blank (local and global options).

The command defaults to clear all options.

EXAMPLE:

Q = clear all modes.

QBN = clear blank and no list mode only.

4.4.13 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 non-alphanumeric delimiter must be used in the command.

Format: S/String1/string2/[^W_F][B][L][N][P][S][D][^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.

W

'steps through' each occurrence of the matched string in the specified range and gives the SXD prompt. This prompt 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 changes only the first instance in every line in the range.

The option default changes every instance of the string within the range.

B

multiple blanks treated as one blank in search and in SUBSTITUTE (local and global option).

L

delimited string option (local and global option). See Y-DELIMIT command.

N

no listing of result (local and global option).

P

print on the 'L0' device (local and global option).

S

suppress listing multiple blanks (local and global option).

D	after substitution, delete remaining characters on the line.
A	all lines in the edit buffer (equivalent to=B,E).
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, will search only the current line.	

EXAMPLE:

S/123/ABC/	= Search the current line. Substitute all occurrences of 123 with ABC.
S/123/ABC/A	= Substitute ABC for 123 in all lines, all occurrences.
S/123/ABC/1,20	= Substitute in line number 1 through line number 20 all occurrences of 123 with ABC.

Option Combinations:

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

S/123/ABC/AW = Step through the entire edit buffer, stopping at each occurrence of 123. The user is prompted by <SXD>. The S option causes the substitution of ABC; the D option gives the prompt <DSS>. Characters entered after the prompt and until the carriage return comprise the new string. The X option ends the command and returns the program to the <EDT> prompt. After the carriage return, the Source Editor completes the substitution and displays the next occurrence.

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

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

4.4.14 UNBANK: Insert the Bank Buffer Records

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 first unbanked line becomes the current line.

Format 1: $U[I] \begin{bmatrix} D \\ K \end{bmatrix} [,line1]$

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, buffer is cleared. Using the K option, a user can unbank the same buffer several times.
- D clears the bank buffer without inserting the bank into the edit buffer. 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.

EXAMPLE:

UI = insert buffer immediately before the current line.

UK10 = unbank buffer, but keep it, and insert it directly after line number 10.

Format 2: $U[D]$

EXAMPLE:

UD = clear the bank buffer without insertion.

4.4.15 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. 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's buffer is considered to be a valid delimiter. Each alphanumeric character not in the buffer is considered to be an invalid character delimiter.

- b. Each non-alphanumeric character in the delimiter's buffer is considered to be an invalid delimiter, and each non-alphanumeric 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.

Format: Y/chars/[I,D]

where:

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

All Y-DELIMIT commands list the current contents of the delimiter's buffer.

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

EXAMPLE:

- Y = list current delimiters
- Y/ABC = clear delimiter's buffer and insert ABC. Then, the command F/XYZ/L would find AXYZ $\emptyset$ or $\emptyset$ XYZB as a match but not WXYZ $\emptyset$. ($\emptyset$ is a valid char, since it is not in the buffer.)
- Y/EF/I = keep contents of buffer and add E and F.
- Y/ACE/D = remove characters A, C, and E from the delimiter's buffer.
- Y/*/I = add "*" to the delimiter's buffer. The command F/ABC/L would find =ABC $\emptyset$ as a match, but not *ABC $\emptyset$.

4.4.16 Z: Backup One Record

The Z command causes the current line pointer to move to the line immediately preceding the current line. (If there are no lines in front of the current line, the current line pointer does not move.)

Format: Z[N][P][S]

where:

- N no listing of results (local and global option).
- P print on the 'L0' device (local and global option).
- S suppress listing of multiple blanks: list as one blank (local and global option).

EXAMPLE:

Z = back up one record.

4.5 THE EDIT BUFFER DATA TRANSFER COMMANDS

The following commands are used to copy data into the edit buffer and out to the source output medium. The NEXT command adds to the convenience of data transfer by allowing the user to copy the edit buffer to the source output device and read a new section of the Source Input device with one command. The commands are listed in alphabetical order.

4.5.1 READ: Copy Source Text Into the Edit Buffer

The READ command copies records from the input file (SI) to the edit buffer. The G option causes records to be ignored and excluded from the new file. Otherwise, the records are stored consecutively in the edit buffer.

The record n option allows the user to select the records to be input. The 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.

Each record 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.

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

Format: R[G][recordn]

where:

- G records are ignored and excluded from new file.
- recordn specifies the maximum record number desired to be read into the edit buffer: default is the last record of the file.

EXAMPLE:

R = copy SI to edit buffer until input limit or end of file.

R30 = READ into the edit buffer the records following the last record read from the SI through record number 30.

RG30 = ignore current record in SI through record 30; (next READ will begin at record number 31).

4.5.2 WRITE: Write Records 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 no listing of results (local and global option).

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.

EXAMPLE:

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.

4.5.3 NEXT: NEXT Section of SI Written 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, 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][,recordn]

where:

G "garbage": records are not put into the edit buffer, they are skipped.

recordn refers to the 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 = input results in error. Record 5 has been read, so record 4 is inaccessible.

4.6 THE EDIT TERMINATION COMMANDS

The following commands are used to terminate the Source Editor and close the files used during the edit process.

4.6.1 TERMINATE: Ending the Editing Process

The TERMINATE command is the final command entered during an editing process. This command creates the edited and renumbered 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 Standard Input device (SI) to the Standard Output device (S0). 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.

The variations of TERMINATE are explained in the description of these formats of the EDIT invocation:

Editing one file with the contents of a second file--Section 3.1.3

Creating a new file from the terminal--Section 3.1.4

Creating a new file from one or more files--Section 3.1.5

Editing several files at one time--Section 3.1.6

When the renaming process is complete, the Source Editor returns control to the OS4 interface.

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. The symbol for this command is the carat, '^'. The user must press the control key to enter the carat. The Source Editor returns control to the OS4 interface with the original file or files intact.

Format 1: T[G]

Format 2: ^

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.

EXAMPLE:

T = terminate the edit process

TG = keep only contents of the S0 and the edit buffer and ignore remaining text in the SI.

^ = terminate the edit process and leave file in its original state.

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 while using the Source Editor.

ERRORS

*** BUFFER CAPACITY ERROR ***

If the input to the bank buffer exceeds its limit, this message is displayed.

*** DEVICE ERROR ***

If a hardware error occurs on one of the assigned devices, this message is displayed.

ERROR IN RENAME --- CHECK FILES

This message is generated if OS4 tries to rename a file with a file name that already exists.

*** LINE # ERROR ***

If the user specifies an illegal line number, this message is displayed.

*** SYNTAX ERROR ***

This message is generated if an error in the format of the command is detected.

?????

When the source text file listed in the EDIT invocation cannot be located on the given device, this message is generated.

WARNINGS

EDIT BUFFER EMPTY

This message is displayed when all records have been removed from the edit buffer and an attempt is made to execute a command on data in the edit buffer.

*** INVALID IN CREATE MODE ***

If Create Mode is in effect, and the user issues a command which is valid only in Edit Mode, this message is generated.

*** INVALID IN EDIT MODE ***

If Edit Mode is in effect, and the user issues a command which is valid only in Create Mode, this message is generated.

*** VALID ONLY IN BLANK MODE ***

This message is displayed if during a Substitute command, the user enters a search string (string1) that consists only of blanks, and Blank Mode is not set. This operation is valid only in Blank Mode where any blanks match any other blanks.

MESSAGES

*** BANK BUFFER EMPTY ***

If an Unbank command is attempted when the bank buffer is empty, this message is displayed.

BYTES AVAILABLE

This message is displayed when the user issues the E command asking for the available memory in bytes.

CURRENT GLOBAL OPTIONS

This message is displayed when the O command is issued, followed by the modes that are in effect.

*** EDIT BUFFER FULL ***

This message is generated when inserted lines fill the edit buffer to its maximum capacity.

EDT...

This message 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

This message indicates that the edit buffer pointer is currently past the last line of buffered data.

*** END OF FILE ***

If the current line is the last record in the edit buffer and an attempt is made to go beyond it, this message is given.

END OF FILE REACHED

If the end of file is reached before the requested line number in a Read command, this message is generated.

INPUT BUFFER LIMIT REACHED

If the edit buffer is 75 percent full before end of file or the requested line number is reached, this message is generated.

THESE ARE NOT DELIMITERS

This message is displayed when the Y command is issued, and gives the characters that are not allowed as delimiters for a matched character string.

REQUESTED LINE # REACHED

This message is displayed on a Read command when records from the current line to the specified line have been copied to the edit buffer.

THESE ARE VALID DELIMITERS

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

LINE TRUNCATED TO 132 CHARACTERS

This message is displayed when more than 132 characters are input in Insert or Input mode with a carriage return <CR>.

LINE OVERFLOW ABORTED

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

APPENDIX B

FORMAT COMPARISON BETWEEN E100 REVISION AND F100 REVISION OF THE SOURCE EDITOR

In the following formats, the pound sign, "#" refers to the line number.

E100	COMMAND	F100	SECTION NUMBER
A	Align	A	4.4.1
--	Bank	B[K][, #1[, #2]]	4.4.2
C[L,N], /S ₁ /S ₂ /[, #1]	Change	S/S ₁ /S ₂ /[^W _F][B][L][N][P][D][S][^A _{#1[, #2]}]	4.3.13
D[L,N][, #1][, #2]	Delete	D[N][R][, #1[, #2]]	4.4.3
F[L,N], /S ₁ /[, #1]	Find	F/S ₁ /[^A _W][B][L][N][P][, #1[, #2]]	4.4.5
G[K,N][, #1]	Goto	G[N][P][, #1]	4.4.6
I[I][C][, #1]	Insert	I[I][, #1] ¹	4.4.7
K[, #1]	Skip	RG[, #1]	4.5.1
L[S][, #1][, #2]	List	L[^V _W][P][S][^A _{line1[, line2]}]	4.4.8
L[A,N]	List [global]	ON,QN	4.4.9
M	Memory	E	4.4.4
--	Move	M[I][line1[, line2],]dline	4.4.10
N	NEXT	N[G][, recordn]	4.5.3
O[E,C]	Mode	O[B][C][N][P][S][L]	4.4.9
--	Print	P[No. of spaces]	4.4.11
--	Quit	Q[B][C][N][P][S][L]	4.4.12
R[, #1]	Read	R[G][recordn]	4.5.1
S[L,N], /S ₁ /S ₂ /[, #1]	Substitute	S/S ₁ /S ₂ /[^W _F][B][L][N][P][D][S][^A _{#1[, #2]}]	4.4.13
T	Terminate	T[G]	4.6.1
--	Unbank	U[I][^D _K][, #1] or UD	4.4.14
W[S][, #1]	Write	W[K][N][, #1[, #2]]	4.5.2
X	Abort	^ [ASCII code : 5E]	4.6.1
--	Y-Delimit	Y/chars/[D,I] or Y	4.4.15
--	Backup 1	Z[N][P][S]	4.4.16
CR	Forward 1	Carriage return	

¹ The C option can be emulated by typing a '?' [ASCII code : 3F] in response to an insert prompt at any point during the insert.

