Time Sharing Operating System (TSOS)

File Editor Reference Manual

May 1971
DJ-003-2-00

Marketing Publications
Building 204-2
Cherry Hill, N.J.

RCA

First Printing: February 1969 (70-00-623)
Reissued: May 1971 (DJ-003-2-00)

PREFACE

This reference manual gives a detailed description of the TSOS File Editor. The TSOS File Editor is a program which creates, modifies, and displays files.

This manual consists of four sections. Section 1 gives an overview of the entire File Editor and its relationship to other software. Section 2 is a comprehensive discussion of the syntax to be used in the detailed descriptions of the commands of the File Editor. Section 3 contains those descriptions and a complete discussion of each of the commands of the File Editor arranged in alphabetic order. Section 4 introduces the user to the File Editor procedure language and its use.

Page

FUNCTIONAL DESCRIPTION

The TSOS File Editor is a program supplied by RCA to enable the user to create, modify, and display catalogued ISAM (indexed sequential access method) files, primarily on an interactive basis. The File Editor is designed to be used conversationally from a terminal so that the user may control processing one command at a time. However, the File Editor may also be used on a nonconversational basis. In the latter case, the commands are either contained in a card deck or have already been stored in a disc file. In any case, the File Editor generally gets its input from a logical sequential file called SYSDTA. The output, in the form of responses and messages, is directed to SYSOUT, which is either a terminal or the printer. (Refer to figures 1-1, 1-2, and 1-3.)

The File Editor consists of basic command verbs, which perform editing functions on the contents of a file. Each verb carries an array of parameters that can specify just what data is to be acted upon by the verb. For example, the verb can be made to operate on a single character within a sequence of lines or it may operate on one entire line. When a verb is presented with its array of optional parameters, it is referred to as a command statement.

A command statement can process a record or a series of records in one file. This file must be cataloged as an ISAM file. Hence, at the beginning of a File Editor session, the user must designate an existing ISAM file or request that an ISAM file be cataloged and made available for the session. The file so designated is then called the principal file.

The File Editor is able to read another file sequentially and makes insertions from the file into the principal file. The File Editor can also write a part of the principal file sequentially into a suitable output file. These input and output files are referred to as secondary files.

The user of the File Editor may collect a series of command statements that are intended for successive execution and store them as a unit in a user file. Such a unit is called a File Editor procedure. A file containing such procedures is called a File Editor procedure file. (A File Editor procedure is comparable to a user-defined macro.)

During a File Editor session, the user may call into operation a procedure with its optional parameters. This has the effect of entering into the session the command statements from the procedure. Each command within the procedure is then expanded with the specified parameter values before its execution. When all the statements in the procedure have been executed, the File Editor returns to the source of the procedure call for its next command statement.

The File Editor has a standard method of handling defaults when the user fails to specify values for the parameters in a command statement. The specific details of default action are given in the Section 2 of this manual. The default method generally consists of carrying symbolic pointers that take on specific values during the session. As each command comes up for execution, it is first scanned. If any commands lack explicit parameters, the current values of the symbolic pointers are used. Other than these symbolics, there is no interconnecting relationship between any two File Editor command statements, with the exception of the JUMP and LOOP commands.

In most cases, each command statement is discarded after execution. There are two significant exceptions: one is the procedure file already mentioned; the other is the LOOP command. The LOOP command allows the user to specify a list of command statements and to execute this list repeatedly until some criterion has been met. The specified list of command statements is stored in the user's virtual memory and then individually executed.

FILE EDITOR PROGRAM STRUCTURE

The File Editor is a Class II program requiring 18 pages of virtual memory for code and seven pages for each user's private work area. The code is reentrant and thus may be shared. The File Editor appears to the TSOS Executive System as a user program.

USAGE

The File Editor can operate on either a conversational or nonconversational basis.

Conversational

In conversational operation, the user activates the terminal by calling and logging on to the TSOS. At LOGON time, the user may specify (within limits) the buffer size associated with SYSDTA; otherwise, File Editor requests a 270-byte buffer. The user must then make the following request to the TSOS control language processor:

EXECUTE (EDIT)

The system then loads the File Editor and types an asterisk to indicate that the File Editor session has begun.

Every input string from an interactive terminal must be terminated by the ETX character, á standard procedure for TSOS.

All File Editor messages and responses go to SYSOUT, which, during a conversational session, is the terminal by definition. If the terminal is a teletypewriter, then an automatic line-feed (↓) and a carriage-return (←) occur before each message and after every 71 characters. The carriage-return and line-feed together are indicated in this manual by the single symbol ←↓. With one exception, explained under the INPUT command, the user can use the ←↓ symbol freely while entering command statements and data from a terminal. Once the user has typed in a command statement and depressed the ETX key, the processing of that statement is under the control of the File Editor. After the File Editor has terminated its processing (successfully or unsuccessfully) and sent responses to the terminal, it requests the next command statement by ←↓ followed by an asterisk, *.

At any time during the File Editor session, the user may break in and interrupt the current activity by depressing the BREAK key. The TSOS control program responds by typing a slash, /, indicating that File Editor processing has been suspended. If any input-output transmission was in progress at the time, the message is discarded. The user may then proceed with any logical TSOS system commands.

To reestablish the File Editor session, the user can issue one of two system commands: /RESUME or /INTR.

/RESUME

This command returns control to the File Editor at the point where it was interrupted except if a message was being transmitted to the terminal when the user pressed BREAK. In this case, the rest of the message is lost, and a new command read is issued.

/INTR

This command returns control to the File Editor's intervention routine, which aborts whatever command was being processed at the time of the break, and displays another asterisk (*) so that the user may issue a new command.

Nonconversational

The File Editor, when operating nonconversationally, can accept input either from the card reader or a sequential cataloged file.

1. A nonconversational session can be entered through the system card reader. In this case, all input lines by definition will be 80 bytes long. The File Editor interprets the trailing edge of the punched card as the end of the line (ETX). Command statements or data must be contained on one card and cannot be continued on another card.

2. A nonconversational task can also be initiated by the system command ENTER, which specifies the name of a cataloged file. The file must have LOGON and LOGOFF as its first and last commands, respectively. An example of a conversational user initiating a nonconversational File Editor session is shown in figure 1-3.

All messages and responses are sent to SYSOUT, which during a nonconversational session is by definition the printer.

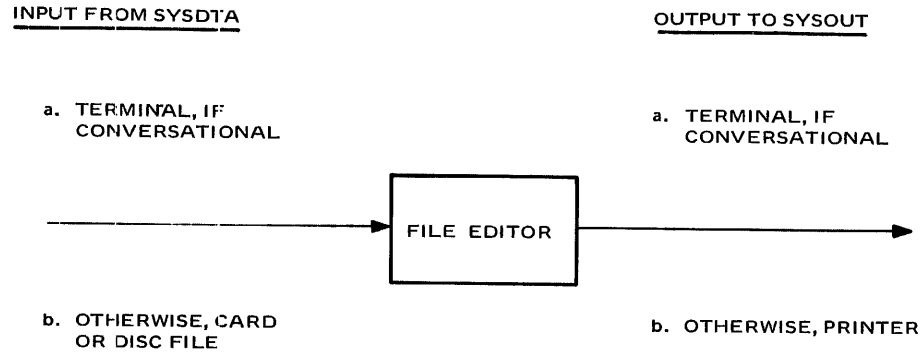


FIGURE 1-1. INPUT AND OUTPUT DEVICES FOR FILE EDITOR

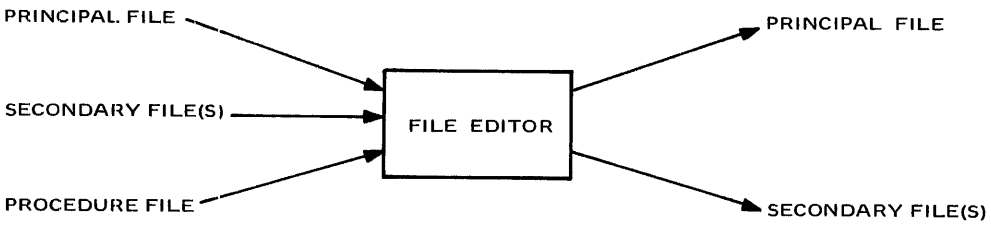


FIGURE 1-2. FILES USED IN A FILE EDITOR SESSION

Explanation	Terminal	Task Y
Conversational session begins.	/ LOGON / EXECUTE (EDIT)	
Terminal is SYSCMD. Task Y awaiting execution.	/ ENTER Y (Task sequence number of task Y displayed by TSOS command processor.)	
Conversational session ends.	/ LOGOFF	
When resources are available, non-conversational File Editor session is executed: SYSDTA=Y		/LOGON / EXECUTE_(EDIT), ΔOPEN X . . . ΔHALT / LOGOFF
During the initial conversational session, execution of task Y is requested by the command ENTER Y. Later, when resources are available, the File Editor is executed under the TSOS Executive with SYSDTA=Y.		

FIGURE 1-3. EXAMPLE OF NONCONVERSATIONAL FILE EDITOR SESSION INITIATED FROM TERMINAL

FILES

Principal File

The File Editor can edit a principal file which is an ISAM file created under TSOS and stored on a random access device; it might have been created in a previous File Editor session, by the DATA command, or by a user program. It may consist of fixed-length (F-type) or variable-length (V-type) records, having numeric keys eight bytes long in any position acceptable to ISAM (refer to the Data Management System Reference Manual). If the file is composed of F-type records, the record length must be 2048 bytes or less; if V-type, the block size must be 2048 bytes.

There can be only one principal file open at a time. If the file is new, File Editor opens it in the OUTIN mode; if old, it is opened either INOUT or INPUT, according to the user's preference.

If a principal file is password-protected, the user must supply that password to the Data Management System either by the PASSWORD command (see DMS Reference Manual), or as one of the parameters of the OPEN command. (See Section 3 of this manual.)

Secondary Files

Secondary input files may be either SAM (sequential) or ISAM (indexed sequential) files, but in either case are opened in the INPUT mode and accessed sequentially. Secondary output files are always sequential (SAM) files and are opened in either the Output or Extend mode, depending on the user's preference.

Lines of a File

A logical record of a principal file is called a line and is the basic unit that the File Editor manipulates. A line consists of an unformatted character string. The characters in the string can be any of the 256 bit configurations, but the string length cannot exceed 2048 characters. A byte of data can be presented to the File Editor in the form of a printable character or a pair of hexadecimal characters; the File Editor can provide output data similarly.

If a file contains F-type records and a smaller record is to be inserted into the file, the record will be space-filled to the right. Records larger than the specified record size will be truncated on the right and File Editor will display an error message.

Line Numbers

To identify each line of the principal file, an 8-digit decimal number is associated with each line in the form of a key. This unique number is passed by the File Editor to DMS for accessing that line.

The user can refer to a line by its 8-digit line number. Optionally, the line number can be expressed to the File Editor in fewer than 8 digits when leading 0's are omitted; it can also have a decimal point to indicate its positional alignment.

The user can also refer to a specific line by means of one of the 10 symbolic line pointers provided by the File Editor. These are named $d, where d = 0, 1, 2. . . . .9. The user can store a line number in one of these symbolic line pointers and then refer to that line in a command by the symbol.

COMMAND LANGUAGE DESCRIPTION

The command language of the File Editor consists of verbs by which the user specifies the editing functions that he wishes to perform on the data in the principal file. During conversational sessions, the commands are entered through the terminal keyboard. During nonconversational sessions, they are entered, typically, on punched cards through the card reader. They may also be stored up during a conversational session and left for later execution by means of the ENTER command as illustrated in figure 1-3. Finally, the File Editor commands may be grouped as a file of commands under the procedure operations to form user-defined macros.

The verbs of the File Editor command language, which fall into four categories, are described briefly below. They are described in detail in Section 3 of this manual.

Control Commands

There are 8 control commands by which a user can open and close a file for editing; can set, reset, and display certain automatic features of the File Editor; and can terminate the editing session.

CLOSE

This command allows the user explicitly to close any file that is open in the File Editor session.

OPEN

This command opens the specified ISAM file as the principal file and will, if necessary, close another current principal file before opening the new file.

HALT

This command closes the principal file and the procedure file, if there is one open, releases the File Editor, and returns control to the command language processor.

HELP

This command displays various messages designed to help the user of the File Editor.

SET

This command sets the desired value into either the current line pointer (*), the symbolic line pointers $d, or the symbolic character counters #, by which the effective domain of the Editor commands may be defined.

RESEQUENCE

This command renumbers the lines of the principal file.

RESET

This command changes the logical switches that define the attributes of a particular session, such as the following:

1. The File Editor's response messages on SYSOUT.
2. Listing of File Editor commands on SYSOUT.
3. The position of the decimal point in line numbers.
4. Internal tabulation settings.

VERIFY

This command causes the current settings of the session-attribute switches and the various symbolic pointers to be displayed on SYSOUT.

Input/Output Commands

The input/output commands provide a means for the user to create new lines in the principal file, to copy lines into the principal file from a secondary file, and to copy lines from the principal file into a secondary file.

TEXT

This command places the File Editor in the text mode, during which lines in the principal file may be created, displayed, deleted, or replaced. The text mode accepts data in character or hexadecimal format.

PRINT

This command prints out a specified portion of the principal file on SYSOUT.

MOVE

This command moves the contents of an existing line of the principal file to a newly created line of the principal file.

GET

This command copies sequentially the contents of a series of lines from a named secondary file into newly-created lines within the principal file.

INPUT

This command enters text from SYSDTA into the principal file.

WRITE

This command copies a specified portion of the principal file into a secondary file.

Line Content Commands

These commands scan a portion of the principal file for the purpose of modifying specific contents.

FIND

This command scans each line in a specified range of lines to locate a prescribed string.

SEARCH

This command searches a specified range of the principal file and displays all instances of the prescribed string.

DELETE

This command deletes a specified portion of the principal file.

ALTER

This command searches an area of the principal file for a specified string and substitutes a given string for one or all occurrences of the specified string.

UPDATE

This command inserts a string of characters at a given point within each line of a given range of lines.

CHANGE

This command permits the interactive user to perform character-by-character editing on a given line or lines.

Special Commands

There are five special commands in the File Editor.

JUMP

This command changes the sequence of execution of commands either conditionally or unconditionally.

LOOP

This command defines a series of commands that is to be executed repeatedly until some criterion is met.

QUALIFY

This command closes the current procedure file, if any, and opens the specified file as the source file for procedure definitions.

NOTE

This command prints the entire contents of the statement in which it appears.

Procedure Call

This command calls a predefined procedure.

RESPONSES AND MESSAGES

The File Editor gives the user two different types of response messages: one is called a diagnostic response message and the other, an execution response message. In the diagnostic response, some flaw has been noted in the relationship between the Data Management System (DMS) and the File Editor or between the user and the File Editor. Execution responses come to the user from the File Editor only; there is no direct communication between the user of the File Editor and the DMS. See figure 1-4.

Diagnostic Messages

Diagnostic response messages indicate, in general, two different classes of error that can occur:

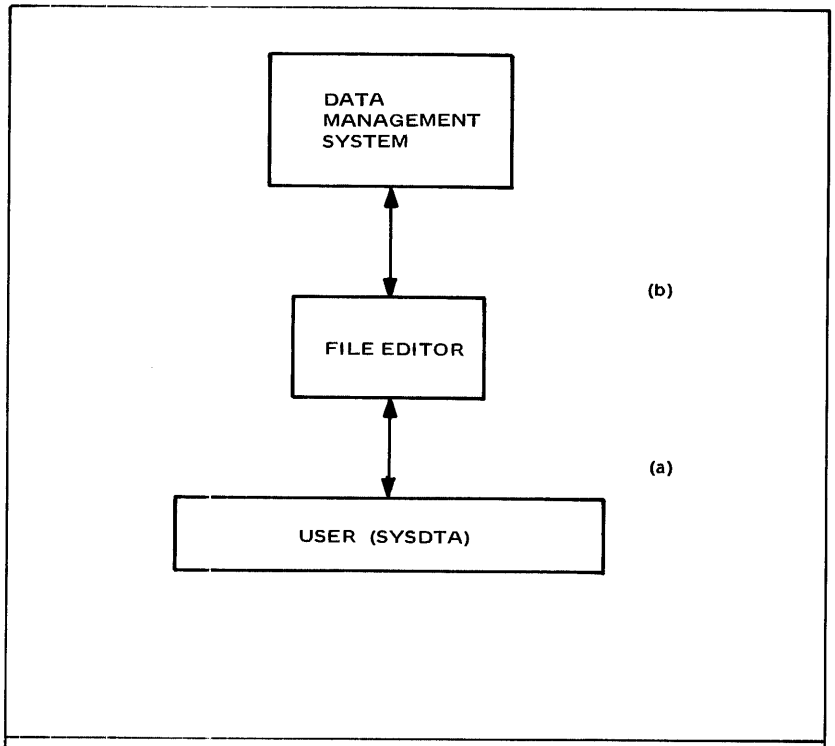
1. Syntax errors, which usually result in two-line error messages.

2. Execution errors, which always result in one-line error messages.

These error messages can occur because the Editor's processing of a command takes place in two phases: an interpretation (or syntax-scanning phase) and an execution phase.

If, during the interpretation phase, the Editor discovers a syntax error in the user's command (for example, unrecognizable operand, command not found, or illegal character), the syntax scan stops, execution is never attempted, and a diagnostic message is issued.

If the syntax scan reveals no errors, the Editor attempts to execute the command as stated. During execution, nonsyntactic errors may be caught, either by the File Editor or by DMS (the execution of most File Editor commands involves interaction with DMS). Certain errors (for example, principal file not open; desired record does not exist) will be revealed before any actual processing on the file is done. Other errors may not appear until after a portion of the command has been executed (for example, an INPUT command, putting records into the middle of an existing file, might try to replace an existing record; and so forth.) In cases like this, any changes already wrought upon the file remain; the command is aborted, and an error message appears on SYSOUT.



The user can communicate directly with the File Editor at (a) and the File Editor can communicate directly with the Data Management System at (b), but there is no direct communication between the DMS and the user of the File Editor.

FIGURE 1-4. COMMUNICATION FLOW

Execution Response Messages

The second major category consists of File Editor messages which are printed during or after command execution. This category is further divided into:

1. File Editor-oriented messages.

2. User-oriented messages.

File Editor-Oriented Messages

File Editor-oriented messages are produced as a result of the successful execution of the commands HALT, OPEN, and QUALIFY and are not under the control of the user. These messages indicate the status of a file after execution of the given command.

User-Oriented Messages

User-oriented messages are further subdivided into:

1. Conversational (TEXT, CHANGE) and output (PRINT, VERIFY) messages.

2. Review messages.

3. Nonconversational messages.

CONVERSATIONAL/OUTPUT MESSAGES

The details of conversational/output messages are described in Section 3 under commands CHANGE, NOTE, TEXT, PRINT, and VERIFY.

REVIEW MESSAGES

Review messages can be of three degrees of complexity:

1. No review – gives no indication of successful execution.

2. Partial review – in general, indicates the line number, length of lines edited, and the total number of those lines.

3. Full review – displays, along with the line number and length of the line, the contents of each line after it has been edited. Upon successful termination of the File Editor command, a count of the total number of lines edited is displayed.

Unless otherwise instructed, the File Editor gives no review information.

In order to obtain either a partial or full review, the user must issue a RESET command indicating the desired degree of review. That degree of review then holds for all further commands until another RESET command is given, calling for a changed degree of review.

The File Editor allows the user the option of receiving on SYSOUT a source copy of the File Editor command currently being executed. This feature is useful primarily to the nonconversational user because it allows him to follow the sequence of commands in execution. This option is called List and is controlled by the RESET command.

If the List option is exercised in an interactive environment, the File Editor will repeatedly display on SYSOUT each user-issued command statement. However, it can be useful during conversational operation to trace specifically the sequence of commands that have been executed in the Loop and Procedure modes.

The NOTE command will always send its output to SYSOUT, and hence the List option does not affect the NOTE command.

FILE EDITOR SYNTAX

Each of the File Editor commands, discussed in detail in the next section, provides the user with the ability to adapt each command to a great variety of different editing operations. A typical editing command may, for example, operate on a single line, a part of a line, on a single character, or upon particular syllables in a whole succession of lines. This section is a general explanation of the symbolism used in the presentation of the specific command.

File Editor Command Statements

A File Editor command statement consists of three main parts:

1. An optional name field followed by at least one space.
2. An operation field.
3. An optional operand field preceded by at least one space.

Spaces and Delimiters

One or more spaces separate the fields from each other. The symbol Δ (delta) is defined as meaning one or more spaces.

Similarly, except in the case of procedure calls, any number of delimiters (equal signs, commas, or spaces) are allowed between operands. The symbol ⫽ (slash ⫽) is defined as one or more delimiter symbols.

⫽ means $\left\{ \begin{matrix} \_ \\ = \\ , \end{matrix} \right\} \left[\left\{ \begin{matrix} \_ \\ = \\ , \end{matrix} \right\} \right] \ \ldots$

The symbols Δ and ⫽ appear in all File Editor commands; they are called operand delimiters.

Name Field

If a command statement is given a name, that name appears in the first field of the statement. It may consist of from two to nine characters; the first character must be a period (.), and the second character must be alphabetic (one of the letters A through Z). The remaining characters, which are optional, may be alphanumeric (A through Z, 0 through 9 only); the name field must be separated from the operation field by at least one space.

Operation Field

The operation field of a command contains either a File Editor verb or a procedure name and is followed by at least one space. The File Editor verbs may be written in full or they may be abbreviated. Thus, the command

OPEN

can also be written

O — The abbreviation for each command is shown in Section 3.

All verbs must be terminated by at least one space if there is an entry in the operand field. If no operands are present, the ETX can immediately follow the command.

Operand Field

The operand field is optional; if present, it must follow the operation field. The operand fields of the File Editor commands may contain a great variety of parameters. By means of these parameters, the user can select and specify exactly what data he wishes the command to operate upon. Each command also carries a complete set of default options that are exercised whenever the user fails to indicate parameters specifically. The general applications of default options are outlined in this section; their specific applications to particular commands are discussed in section 3 of this manual.

Some commands carry more than one operand parameter; in such cases each parameter must be followed by at least one delimiter. Except for the RESET and VERIFY commands, the parameters carry positional significance; thus, the exact order shown in the symbolic form of the command must be observed. In the case of an interactive user at a terminal, the last parameter must always be followed by an ETX to initiate execution of the command.

OPERAND SYNTAX

The domain of a File Editor command is that portion of a file in which the command is to become effective. The domain of a command is defined by two types of parameters:

1. Line address parameters that indicate a range of lines.

2. Line content parameters that indicate a specific segment of every line.

In the syntax of each command in this manual, line address parameters are indicated by the symbols a1 and a2; line content parameters are indicated by Ce1 and Ce2.

A keyword parameter specifies a unique situation within the domain of a command. This type of parameter is usually last in the operand field and, if used, may usually be abbreviated to its first letter. For example, the keyword parameter OLD can be written also as O, and the keyword NEW, as N.

If a keyword parameter is not specifically stated, a default value is used in the execution of the command. The default values are indicated in the description of each command by the underline.

The entire input command statement is scanned syntactically, and the value of each parameter is determined before the command is executed.

Line Address Parameters

Line address parameters are denoted by the symbols a1 and a2; the first symbol indicates the address of the line at which the operation is to begin, and the second symbol indicates the address of the line at which the operation should end. There are five different ways of assigning values to a1 and a2. They are:

1. Line number.
2. Current line pointer.
3. Last line pointer.
4. Symbolic line address.
5. Relative line address.

Line Number

A line number (also known as key) may consist of one to eight decimal digits that uniquely address a line of a file. Although the numbers are stored internally as full 8-digit numbers without decimal points, the user may instruct the File Editor to introduce a decimal point anywhere in the line number.

Thus, if the user wishes to set the decimal point between the third and fourth digit from the right, he would enter the command:

RESET P3

The effect of the user's specifying an assumed decimal point is shown in figure 2-1. The number 1.2476 is illegal because there are more than three digits to the right of the decimal point and thus violates the value assigned in the RESET command.

User	Internal
1	00001000
1.2	00001200
8081.247	08081247
1.2476	Illegal

If the user issues the command, RESET P3, the File Edtior will accept all the above numbers except 1.2476, even though all numbers appear inside the computer as shown at right. The last value on the left carries four digits to the right of the point, instead of three or less, and is therefore unacceptable.

FIGURE 2-1. LINE NUMBER RELATIONSHIPS

Current Line Pointer

The current line is defined as that line which will next be processed in default of other explicit line address specifications. The symbol designating the current line number is the asterisk *. If a command has a domain represented by a1 and a2, then the value of * will point to the next line to be processed, if the command is effective. When the command has processed its final line, a2, the value of * will be set equal to a2. The convenience of the current line pointer arises when the values of the line address parameters are not specified; the default value of * will be used to define the domain of command execution.

Notes:

1. When a principal file is first opened, the value of * is set equal to the first existing line of that file.

2. If a File Editor command modifies the value of * the way it does so is described in the discussion of that command.

3. The user may change the value of * by using the SET command.

For example,

SET __*__=__200

sets the value of * equal to 200.

Last Line Pointer

The last line of the principal file is designated symbolically by $. The use of $ as the a2 parameter would extend the domain of the command to the last line of the principal file. The File Editor automatically adjusts the value of $ as new lines are appended to the file or old lines removed from it.

The expression $-n indicates the nth line from the last line in the file; the expression $+n is meaningless, and thus illegal.

Symbolic Line Pointers

A symbolic line pointer is a symbol whose value can be set to any nonnegative, 8-digit decimal number to designate a line in a file. The File Editor provides 10 such symbolic line pointers by the use of the symbol $n, in which n = 0, 1, 2, . . . , 9. The user may modify the value of $n (n = 0, 1, 2, . . . , 9) with the SET command.

For example,

SET $3 = 250

When the File Editor opens the principal file, it sets all 10 symbolic line pointers equal to the line number of the first existing line of the opened file.

The SET command allows the user to set any $n to any line address parameter, a; it does not require that a line with the address a exist, a feature which is useful for creating new lines.

Relative Line Addressing

If there is a line in a file that can be addressed by a, then the expression a ± e is a relative line address in which e is a numerical value that specifies the number of lines preceding (-) or succeeding (+) line a. In order for the relative line address 1000+5 to be valid, line 1000 must exist and be followed by five lines of the file; 1000+5 then refers to the fifth line after 1000, whatever its actual line number may be. Similarly, for 1000-5 to be valid, line 1000 must exist and be preceded by at least five lines of the file.

Notes:

1. If in relative line addressing the a is omitted, the current line is assumed; that is, +3 is equivalent to * + e.

2. Unless otherwise specified, a ± e is a permissible form whenever the symbols a, a1, or a2 (line address parameters) appear in this manual.

Line Address Formats

Some File Editor commands require a clearly defined range of lines. This range is expressed as a1 ⩔ a2, indicating that the command is to act upon all lines from a1 to a2 inclusive.

Other commands require only one line address parameter. The syntax of this type of command uses either the letter a or ℓ. The a represents the address of a line which may or may not already exist in the file. The ℓ represents the address of a line which cannot exist. Although the details of the a and ℓ are given in the respective command descriptions, their purpose can be summarized with several of the File Editor commands.

The Get, Input, and Move commands, for example, all obtain data from a secondary source and insert them into the principal file starting at line address ℓ. The requirement that ℓ should not currently exist in the file assures the user that he will not inadvertently destroy existing data. If several lines are inserted and one of the later line numbers should coincide with or exceed one already in existence, the insertion process stops automatically before destroying or overstepping the existing line.

In other words, insertion of lines must begin at a new line number and stop before the next higher existing line number.

The CHANGE command allows the user to change the contents of a specific line. Hence, the line address parameter is stated as the symbol a and must be an existing line.

Finally, the TEXT command offers the user the opportunity to create new lines as well as display, delete, or replace existing lines of the principal file. The line address specified by the symbol a in these various cases may or may not exist, depending upon the needs of the command.

Defaults

The line address parameters are expressed within square brackets to indicate that they may remain unspecified if the user so desires. Although the detailed default actions are listed in the individual command narratives, the following default actions, which are consistently applicable to all File Editor commands, may be considered at this point.

Defaults of a1 [Ø a2]

1. If a2 is not specified, then a2 is set equal to a1.

2. If neither a2 nor a1 is specified, both are set equal to *.

Defaults of [a]

In the CHANGE and SET commands, the value of * is used when there is no value assigned to [a]. If the parameter a is omitted in a TEXT or INPUT command, the value of $ (the last line in the file) is incremented by the standard or given increment, n, for each new line added; this new value of $ is used in default of a.

Default of [ℓ]

If ℓ is not specified in the GET, INPUT, and MOVE commands, the same default action described for default of *a* in TEXT is used; that is, the newly created lines are appended at the end of the principal file.

Line Content Parameters

The line content parameters provide the user with the means to manipulate the contents of lines or line substrings. The line content parameter, therefore, is a command parameter that specifies the location of a substring within a line; a substring may also be an entire line.

There are two types of line content parameters.

1. The string parameter identifies a group of successive characters within a line.

2. The character count parameter identifies a position within a line.

String Parameters

The string parameter called for by some File Editor commands may be composed of a string of up to 255 characters enclosed in apostrophes (') or quotation marks ("). The string parameter may also be an address of a line whose content is the string parameter. The string parameter is written symbolically as

s;

in the user's input, it may take one of three forms:

$\left\{\begin{matrix}'\\ ''\end{matrix}\right\}$ string $\left\{\begin{matrix}'\\ ''\end{matrix}\right\}$

in which string can contain up to 255 characters;

$\left\{\begin{matrix}'\\ ''\end{matrix}\right\}$ string $\left\{\begin{matrix}'\\ ''\end{matrix}\right\}$

in which the string can contain any even number of hexadecimal characters up to 510; and

La

in which the string referred to is the complete contents of line a in the file.

A character string may be bounded by apostrophes or quotation marks. In a command statement where the File Editor is accepting a string as input, for the CHANGE or UPDATE, the string is always enclosed in apostrophes. In a command statement which directs the File Editor to search for a particular string, for instance the FIND or ALTER, the choice of apostrophes or quotation marks to enclose the character string allows the user to specify more precisely the required searching operation. This is achieved through the recognition of text delimiter characters. By means of the RESET command, the user may define any characters to be text delimiters. The set of 256 characters can thus be divided into two mutually exclusive classes: text delimiters and those characters that are not text delimiters. Unless otherwise instructed, the File Editor considers the following characters to be text delimiters:

(; . " + /) : ! ? - sp , ' = *

Consider the example,

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	P	I	T	E	S	T				C	L	I				T	E	S	T	,	X	'	O	R

assuming the default case of text delimiters to be in effect. The string of characters, TEST, appears twice in the line of text. In the first occurrence it is preceded by a nondelimiter character.

In specifying a string, the user employs symbols called string sensors. In the process of describing a particular string, the user may realize that perhaps it will not always have delimiters around it. Therefore, the File Editor has two symbols called string sensors. These symbols are the apostrophe (') and the quotation mark (").

The apostrophe (always written ') indicates that a text delimiter character must be present at the indicated point. The quotation mark (always written ") indicates that a text delimiter may or may not be present. A character string must be preceded and followed by one or the other of the string sensor symbols.

Suppose that a range of lines is to be searched for the string ABCD. If the user enters 'ABCD", he would thereby indicate that the string he wishes to find must be prefixed by a text delimiter, and it may or may not be suffixed by a text delimiter.

If an apostrophe or quotation mark also exists in the string, as a character, it must be duplicated to prevent ambiguity.

Thus,

'AREA' 'X' names the string AREA'X.

Assume that the class of text delimiters has been reset to: space ( ), plus (+), and comma (,) and the current line is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
P	I	S	W	T	A				M	V	I				S	W	T	A	+	1	,	X	'	

In a searching operation, the string of characters SWTA can be described in the ways shown in figure 2-2.

String Parameter	Meaning
'SWTA'	The string SWTA must be preceded and succeeded by a text delimiter. Only one match is made at column 16.
"SWTA'	The string SWTA may be preceded but must be followed by a text delimiter. Two matches occur; at columns 3 and 16.
'SWTA"	The string SWTA must be preceded and may be succeeded by a text delimiter. Only one match at column 16.
"SWTA"	The string SWTA may be preceded and succeeded by a text delimiter. Two matches occur at columns 3 and 16.
Through proper indications by means of the quotation mark and apostrophe, any given string of characters may be isolated from surrounding text.	

FIGURE 2-2. STRING PARAMETERS

Character-Count Parameters

The character-count parameters define the domain of a command as a fixed field throughout a range of lines. A single parameter points to a unique character position in a line. A pair of character-count parameters specifies two unique character positions within a line, and thus defines an inclusive field within that line. A pair of character-count parameters in conjunction with a range of line numbers consistently defines a field in every line of that range.

A symbolic character-count parameter is a symbol whose value can be set to any nonnegative integral number not exceeding 32K. There are 10 symbolic character-counters that are represented as:

#d where d = 0, 1, 2, . . . , 9.

The values of these counters are defined at principal file initiation by the OPEN command which sets each #d (d=0, 1, 2, . . .,9) equal to 1. The user may modify these values with the SET command.

Certain File Editor commands preempt the user's control of the values in certain symbolic parameters; they use #0, #1, and #2 as pointers to indicate the outcome of a completed command. The commands which affect these pointers are:

1. The commands that affect only #0 are:

ALTER	GET	MOVE	WRITE
DELETE	INPUT	UPDATE	

2. FIND affects #0, #1, and #2.

3. OPEN affects #0 through #9.

Note: The symbolic character counters #d (d = 0, 1, 2, . . . , 9) are considered halfword binary counters. The user can perform positive fixed-point arithmetic with these variables. For example, if the value of

#5 is 35, and

#7 is 20

then the expression

#5-#7+2

will have the value 17.

Current Line-Size Symbol

The number sign (#) represents symbolically the total number of characters in the current line. The value of # cannot exceed the maximum record size. Because the value of # is a function of the current line, it may not be changed by the user. If the user changes the value of * and thus points to a different line of the file, the value of # is automatically adjusted so as to reflect the size of the line at the new value of *.

Note: The symbol # is similar to #d in that it may be considered as an operand in positive, fixed-point arithmetic. For example, if * is pointing to a line having 200 characters in it, and if #6 has the value 15, then the expression

#-#6+5

has the value 190.

Example – Suppose that the following conditions prevail:

* is set to line 50

#4 = 6, #1 = 19, #2 = 26

and line 50 consists of 36 characters:

THIS IS A LINE OF THE PRINCIPAL FILE.

The graphic representation of line 50 is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
T	H	I	S		I	S		A		L	I	N	E		O	F		T	H	E		P	R	I	N	C	I	P	A	L		F	I	L	E	

Notice that C# represents the 36th position, since # is the number 36. Line position 4 may be pointed to by #-#2-#4.

Other special uses of the # symbol are described under the JUMP command.

Individual Character-Count Parameters

An individual character-count parameter has the symbolic form, Ce, in which the e is an arithmetic expression. Every character-count parameter must be capable of being resolved into a positive integral number not greater than 32K. The value of the parameter points to the nth character position of the line, where n is the value of the positive integral number.

Arithmetic Expressions

Any of the following are valid arithmetic expressions:

1. A decimal integer not exceeding 32,000.

2. The current line size symbol, #.

3. A symbolic character counter, #d (d = 0, 1, 2, . . . , 9).

4. An arithmetic expression of the form:

P1 ± P2± . . . ± Pi in which any element Pj can be a decimal integer, current line-size symbol, or a symbolic character counter.

Character Fields

Some File Editor commands call for the definition of a character field. The syntax which defines a character field is written as a pair of character count parameters:

. . . Ce1 [Ⓟ Ce2] . . .

The first character of the field is pointed to by Ce1 and the last character by Ce2, with the relationship e1 ⩽ e2 understood.

Either or both of the character count parameters may be omitted. In default of one or more such parameters, all File Editor commands take the following actions:

1. If only one parameter is specified, it is assumed to be Ce1. The File Editor assumes that the field extends from Ce1 to the end of the line.

2. If both parameters are missing, the File Editor assumes that the field extends across the entire line.

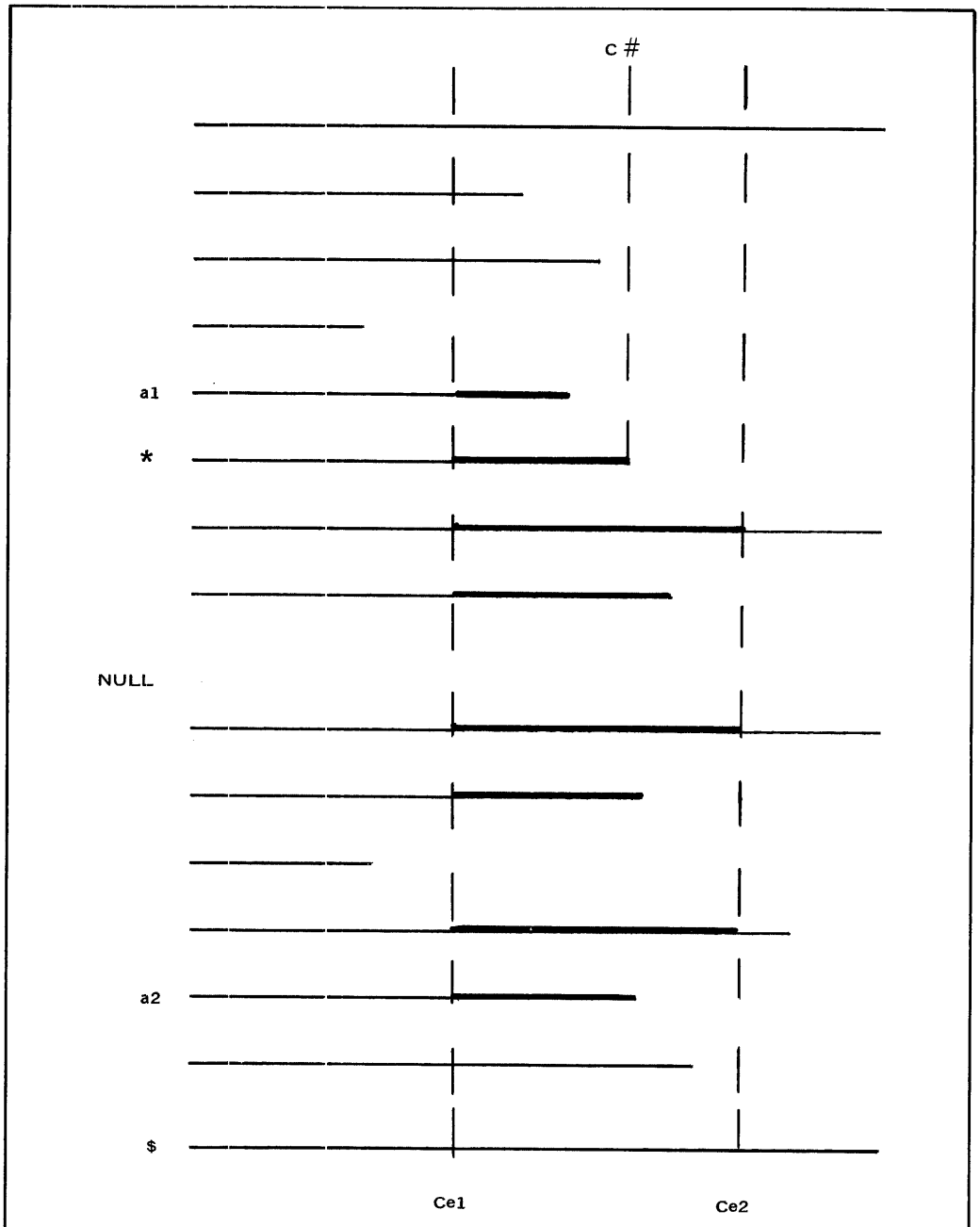
3. If the requirement that e1 ⩽ e2 is violated, the command is aborted and a diagnostic message is given on SYSOUT.

4. If any line within the specified range does not contain all the character positions indicated by Ce1 or Ce2, then part or none of that line will be edited. The following principles govern the editing in such a case:

   a. If a line has fewer characters than specified by Ce1 (and hence Ce2), the line is considered outside the domain of the command and no editing takes place.

   b. If the line has more characters than Ce1 but fewer than Ce2, the field to be edited extends from Ce1 to the end of the line.

Example – The domain of a command is defined by the parameters a1, a2, Ce1, Ce2. Graphically, the domain is shown by the heavier lines in figure 2-3.



The horizontal lines represent the lines of a principal file. The line address parameters, a1 and a2, define the domain or range of effectiveness of a command. The character count parameters, Ce1 and Ce2, define the range within the lines. The lines affected are drawn with heavier-weight lines. The current line pointer is indicated by the asterisk, *, in the line below a1. The value of C # is determined by the length of the line currently in process. The dollar mark, $, indicates the last line in the principal file.

FIGURE 2-3. DOMAIN OF A PRINCIPAL FILE

Session Attributes

The particular attributes of any session with the File Editor are determined by the values assigned to a group of variables that can be controlled by the RESET command. In all cases, there are defined default attributes that are in force at the beginning of the session. The following discussion covers both the options offered by the RESET command and the default attributes supplied by the File Editor.

Text Delimiters (D)

The default set of text delimiters has been previously described in the discussion of string parameters. The RESET command grants the user the right to make any set of the 256 available characters into text delimiters.

Guard Characters (G)

A point of possible confusion exists when a user wishes to create a program file to be used in a nonconversational session as SYSDTA or SYSCMD. If this file is to have system commands, the user is confronted with the problem of inputting a line whose first character is a slash (/) without having the command interpreted as a system command for immediate execution. To resolve this confusion, the File Editor designates one character as the guard character.

The guard character is a unique character that may precede a line of input when using the INPUT, TEXT, or CHANGE commands. The guard character is deleted by the File Editor and the subsequent input is interpreted purely as text. It is not interpreted as a File Editor command or as a system command.

The guard character for the File Editor in event of default is the period (.). The user is cautioned against using the / or ˜ characters as guard characters.

For example, the TEXT command allows the user to input data through the terminal. The user indicates the end of his data by issuing the subcommand, #END. If, however, the user wished #END to be part of his data (and assuming that the guard character were set to %), he would type:

% #END

The guard character % would be deleted and the desired data, #END, would then be stored.

Increment (I)

If the user is creating a series of new lines by a single command, the File Editor will automatically calculate subsequent line numbers provided that the user specifies the number for the first line and an increment, n, from which to calculate succeeding line numbers. Specifically, the number (key) of the ith new line can be calculated from the formula.

k_i=first number + (i-1)n.

If the user wishes, he may specify a temporary line increment, to be used in a given instance of a GET, INPUT, MOVE, or TEXT command. If he does not, the File Editor takes the present value of the standard line increment I. The standard increment I can be reset to any number from 0 to 99999999, inclusive; its default value is 100.

Listing (L)

The listing option allows the nonconversational user to trace the execution of each of the File Editor commands. When this option is invoked by the RESET command, the File Editor displays each command statement on SYSOUT before it checks the syntax of the command and executes it.

Decimal Pointer (P)

The File Editor allows the principal file to have as many as 100,000,000 unique line numbers, each falling between

00000000 and 99999999, inclusive.

The user may set an assumed decimal point in any one of the nine possible positions by the RESET command. The positions are numbered 0 through 8 from right to left. In default of a specific setting, the File Editor assumes P0. In this case, the numbers 00000351 and 351 are equivalent.

Review (R)

The complexity of information supplied by the various degrees of review is described in Section 1. Without special action by the user, the File Editor always gives no review. In order to obtain either partial or full review, the user must issue a RESET command indicating the desired degree of review. That degree of review then continues for all commands until another RESET command is given in which some other degree of review is called for, including the reestablishment of no review.

Tabs (T)

When he is creating new lines in the principal file with the TEXT and INPUT commands, the user may wish to format those lines through the use of tabulation positions. The File Editor allows a maximum of 10 such tabulation positions. Associated with each terminal device is a control character called TAB, which the File Editor interprets as a directive to space-fill to the next tabulation position, if it exists. Otherwise, the tab character is treated like any other input character.

The File Editor supplies four groups of tab settings that the user can request with the RESET command. They are:

1. No tabs.
2. Assembly tabs.
3. COBOL tabs.
4. FORTRAN tabs.

The language tabs are supplied primarily to assist the user in the preparation of source-language program files, but may also be useful in formatting lines of a data file.

This section describes each of the following File Editor command verbs:

ALTER	NOTE
CHANGE	OPEN
CLOSE	PRINT
DELETE	QUALIFY
FIND	RESEQUENCE
GET	RESET
HALT	SEARCH
HELP	SET
INPUT	TEXT
JUMP	UPDATE
LOOP	VERIFY
MOVE	WRITE

ALTER

The ALTER command searches a domain of the principal file for the first or all occurrences of a specified string, and replaces it with another specified string.

Name	Operation	Operand
[Δ][.nameΔ]	{ALTER / A}	[ⅅa1 [ⅅa2]][ⅅCe1[ⅅCe2]] ⅅs1 ⅅs2 [{ⅅFIRST / ⅅF / ⅅALL / ⅅA}] [ⅅ]

a1 and a2

These two line-address parameters define the range of lines of the principal file to be edited.

If a2 is not specified, the line indicated by a1 is edited. If neither a1 nor a2 is specified, the line indicated by the current line counter * is edited.

Ce1 and Ce2

This pair of line-content parameters defines a contiguous portion of every line of the range. Notice that the quadruple a1, a2, Ce1, Ce2 defines the domain of this command.

If Ce2 is not specified, the line is edited from Ce1 to the end. If neither Ce2 nor Ce2 is specified, the entire line is edited.

s1

This mandatory string parameter is the search argument for the command (see Section 2 for a discussion of string parameters). If s1 is null and Ce1 and Ce2 are omitted, then all null lines within the domain are replaced by the string s2. If s1 is null and Ce1 and Ce2 are specified, command execution is terminated and a diagnostic message is displayed.

If neither s1 nor s2 is specified, command execution is terminated and a diagnostic message is displayed on SYSOUT.

s2

This mandatory string parameter is the replacement string for s1. Because s1 and s2 do not have to be of equal length, the replacement string s2 may contract or expand the line. If s2 is null, then s1 is deleted. This process, however, never removes a line from the principal file even if that line becomes null.

FIRST (F)

The keyword parameter FIRST (abbreviated to F) specifies that, after altering the first s1, the command is to terminate. This means that the command, if successful, alters only one line of the file; * will be set to the address of that line.

ALL (A)

The keyword ALL (A) specifies that every s1 within the domain of the command is to be altered.

Command Execution

The ALTER command scans the specified portions of each line of the domain for an occurrence of s1, and if a match is found, a check is made on the string sensor requirements. If these requirements are satisfied, then the string s1 is replaced by s2 and the scan pointer is positioned to the right of s2. If the ALL (A) keyword was specified, the scanning and altering process continues to the end of the domain. Otherwise, the command is terminated.

Command Termination

Indicators:

#0	If at lease one s1 was altered, hen the command was effective and #0 is set to 1. Otherwise, the command was ineffective and #0 is set to 0).
*	The value of * is set to the line address of the last line scanned. (The use of keyword FIRST (F) causes * to be set at the line address where the first s1 was altered.)

Responses:

No review	No output to SYSOUT
Partial review	For every altered line: Line Number Final line length and total count of lines altered
Full review	For every altered line: Line Number Final line length Altered contents and total count of lines altered.

Example

```
* R R, TA
*I 5,I5

*BEGIN MVC  X,Y
       L    REG5,A
       ST   REG7,Y
       B    NEXT

*X     DC   A(ITEM1)
Y      DC   A(POINT)
Z      DC   A(SEQ)
#END
*
R RF
 REVIEW MODE IS FULL
*
 ALTER 5,35,C16, "Y"='Z',ALL
         5.      <18>
 BEGIN     MVC     X,Z
          15.      <21>
           ST      REG7,Z
 NUMBER OF LINES AFFECTED:   <2>
*

R R
*
P 1 $

 BEGIN     MVC     X,Z
           L       REG5,A
           ST      REG7,Z
           B       NEXT
 X         DC      A(ITEM1)
 Y         DC      A(POINT)
 Z         DC      A(SEQ)
 *
```

CHANGE

The CHANGE command edits a line of the principal file character by character.

Name	Operation	Operand
[Δ][.nameΔ]	{CHANGE / C}	[ⅅa][ⅅX][ⅅ]

a

Indicates the line number of the line in the principal file that is to be edited. If a is not specified, then the value of * is used.

X

If specified, indicates that input and output are to be in hexadecimal format. Otherwise, character format is employed.

Command Execution

The line specified by the letter a is displayed on SYSOUT. The user then types in a line of editing information. After reading the line of editing information, the File Editor scans it in parallel with the original line and constructs a new line character by character. Each character of editing information tells the File Editor how to process the corresponding character in the original line. The editing characters are:

– Retain this character.

? – Delete this character.

A quoted string – Insert this string. (See Section 2 for a discussion of string parameters.)

Any character – Replace original character.

ETX – Retain the rest of the line.

File Editor scans and acts upon the editing information one character at a time until it has all been used.

If the hexadecimal mode is set, then the original line is displayed in two-character hexadecimal format. The editing information also is expected in two-character hexadecimal format, thus:

– Retain this character.

?? – Delete this characters.

Quoted hexstring – Insert this string (even number of hexadecimal characters).

Any pair of hex characters – Replace original pair of hex characters.

ETX -- Retain the rest of the line.

The editing information must consist of an even number of characters.

After creating the new line, the File Editor reads the next command from SYSDTA. If this is a null command (ETX), then the CHANGE process is initiated automatically for the next existing line in the principal file. The CHANGE process terminates when a nonnull command is issued from SYSDTA after the creation of a new line.

Command Termination

Indicators:

*	Points to the last line displayed.

Responses:

No review	Content of old line.
Partial review	Content of old line and number and length of new.
Full review	Number, length and content of old and new lines.

Programming Notes

1. A guard character (. in default) may be used when the editing information has a meaningful character at the beginning (/or #).

2. In specifying a string, the user must follow standard string rules. (See Section 2.)

The insertion of a string may result in the new line being longer than the old. In fact, it is the only means by which a changed line can be longer than the old one.

3. In case of error, the command is aborted and * will point to the last line scanned.

Example

```
* R R
D 100
*
INPUT 100
*ABCDEFGHIJKLMN
#END
*CHANGE 100
 ABCDEFGHIJKLMN
*./??'XYZ'##PQRS      (Note: two spaces follow "PQRS".)
*P 100,N
       100.      <15>
/XYZDEPQRS  LMN
*R R
 *D 100
*INPUT 100
*ABCDEFGHIJKLMN
#END
*CHANGE 100,X
 C1C2C3C4C5C6C7C8C9D1D2D3D4D5
*.61????'E7E8E9'####D7D8D9E24040
*
P 100 N X
       100.     <15>
 61E7E8E9C4C5D7D8D9E24040D3D4D5
*
P 100 N
       100.     <15>
/XYZDEPQRS    LMN
*
```

CLOSE

The CLOSE command allows the user explicitly to close any file that is open in the File Editor session.

Name	Operation	Operand
[Δ][.nameΔ]	{CLOSE / CL}	[ΔfilenameΔ*PROCEDURE*Δ*PROC*Δ*PRINCIPAL*Δ*PRIN*Δ*ALL*]

filename

The fully-qualified name of the file the user wishes to have closed.

PROCEDURE
or *PROC*

Either of these keyword operands indicates that the user wishes to close whatever procedure file is open.

PRINCIPAL
or *PRIN*

Either of these keyword operands indicates that the user wishes to close the principal file which is currently open.

ALL

This keyword parameter instructs the File Editor to close the principal file which is currently open and also the procedure file, if there is one currently open.

If the user does not specify any parameter, File Editor assumes *ALL* was intended.

Command Execution

File Editor closes whatever files the user has requested and puts out a message naming the files closed. If the user specifies the name of a file which is not open at this time, or if he specified *PROCEDURE* (or *PROC*) and no procedure file is open at this time, the File Editor issues an error message to draw his attention to this fact. In the same way, a user who gives the command CLOSE *PRIN* when no principal file is open is made aware of the discrepancy.

Command Termination

Indicators: Not affected.

Responses: The names of any files closed are displayed on SYSOUT.

Example

Assume that the principal file being edited is named PAYROLL.A and that the procedure currently in use is named SALARY.DETAIL when the File Editor receives the command CLOSE*ALL*.

```
User:          CLOSE *ALL*

File Editor:   CLOSED PRINCIPAL FILE PAYROLL.A
               CLOSED PROCEDURE FILE SALARY.DETAIL
```

DELETE

The DELETE command deletes a specified domain of the principal file.

Name	Operation	Operand
[Δ][.name Δ]	{DELETE / D}	[⍁a1[⍁a2]][⍁Ce1[⍁Ce2]][⍁s][⍁]

a1 and a2

These two line address parameters define the range of lines in the principal file to be processed. If a2 is not specified, only the line indicated by a1 is edited. If neither a1 nor a2 is specified, the line indicated by the current line pointer * is edited.

Ce1 and Ce2

This pair of line content parameters defines a contiguous portion of every line in the range. Notice that the quadruple a1, a2, Ce1, Ce2 defines the domain of the command.

If Ce2 is not specified, the line is edited from Ce1 to the end. If neither Ce1 nor Ce2 is specified, the entire line is edited.

s

If this parameter is specified, then every string s within the domain is deleted, thus contracting the line where each string s occurs.

Command Execution

If a string parameter s is not specified, the entire domain is deleted from the principal file.

Null lines within the domain will be deleted only if Ce1 and Ce2 are not given.

If s is specified, then the domain is scanned and each s is deleted.

Command Termination

Indicators:

#0	Set to 1 if anything is deleted; otherwise, it is set to 0.
*	Set to the last line processed in the domain.

Responses:

No review	None.
Partial review	List of line numbers and total number of lines affected.
Full review	If the whole line is deleted (neither Ce1 nor Ce2 specified) then the response is the same as for partial review. If lines are changed, then the line number and its new contents are displayed along with the total number of lines affected.

Programming Notes

The command DELETE 1 $ s (where s is a string of three, ten, or more spaces enclosed by apostrophes) is useful for quickly deleting almost all trailing blanks from a file. (Such a file might have been created by a background task from punched cards, using the TEXT command.)

Example

```
R R, TA
*
D 1 $
*
INPUT 5, I5

*COMPRE              CLC     A,B
                     BE      X
                     BH      Y
                     B       Z

*Y                   CLC     A,B
                     BNE     Z
                     B       X

#END
*
R RF
 REVIEW MODE IS FULL
*

DELETE 10,25

  10. DELETED  15. DELETED  20. DELETED  25. DELETED

NUMBER OF LINES AFFECTED:     <4>
*

P 1 $

        5.     <18>
COMPRE     CLC  A,B
       30.     <16>
           BNE  Z
       35.     <16>
           B    X
*
```

Example A

```
* DELETE 1 $

* INPUT 5,I5

*SYMB1   DC          A(TIMB1)
SYMB2    DC          A(TIMB2)
SYMB3    DC          A(TIMB3)

*MMB1    MVC         MB1,SYMB1
         CLC         SYMB2,MB1

#END

*RESET   RF

 REVIEW MODE IS FULL

*FIND 5, $, 'MB1'

        20.          <24>

MMB1     MVC             MB1,SYMB1

 NUMBER OF LINES AFFECTED:      <0>

*VERIFY *, #0, #1, #2

* = 20.

#0 = 1

#1 = 16

#2 = 18
```

Example B

Using ALTER to find all occurrences of string 'THE':

```
*PRINT 1, $, N
    100    <9>
THE QUICK
    200    <9>
BROWN FOX
    300   <15>
JUMPED OVER THE
    400    <8>
LAZY DOG
    500   <19>
AND THE SLEEPY CAT.
*RESET RF
 REVIEW MODE IS FULL
*ALTER 1, $, 'THE', 'THE', ALL
    100    <9>
THE QUICK
    300   <15>
JUMPED OVER THE
    500    <19>
AND THE SLEEPY CAT.
NUMBER OF LINES AFFECTED:       <3>
*
```

GET

The GET command copies a domain of a secondary file into the principal file.

Name	Operation	Operand
[Δ][.nameΔ]	{GET / G}	[Ⱥ {C'n' / X'h'}] Ⱥ filename [{ⱭRe1 [ⱭRe2] / ⱭKk1 [ⱭKk2]}] [ⱭCe3 [ⱭCe4]] {[Ⱥℓ] [ⱭIn] / ⱭKEY / ⱭK} [Ⱥ]

C'n'
X'h

This parameter gives the read password required to open the secondary file. If not present, no password should be required to read this secondary file.

filename

This parameter specifies the name of the secondary file to be accessed. This parameter must be given.

Re1

Re1 is a symbol made up of a key letter R and an expresssion e1, the value of which must be a positive nonzero integer specifying that copying should start with the e1*th* logical record from the beginning of the secondary file.

Re2

Re2 is a symbol made up of a key letter R and an expression e2, specifying that copying should end with the e2*nd* logical record from the beginning of the secondary file. If Re1 is specified but not Re2, then all the secondary file from the e1*st* line up to and including the last line is copied.

Kk1

Kk1 is a symbol made up of the key letter K and eight decimal digits. This symbol specifies the key of a line in the secondary file at which copying is to begin.

> Note: The Kk parameters should be specified only if the secondary input file is either an ISAM file with 8-byte numeric keys or a SAM file in which each record begins with 8 numeric characters.

Kk2

Kk2 is a symbol made up of the key letter K and eight decimal digits. This symbol specifies the key of a line in the secondary file at which copying is to end. If Kk2 is not given and if Kk1 is specified, then all the secondary file from Kk1 up to and including the last line is copied. (See programming note 5 below.)

Re1,Re2
Kk1,Kk2

If none of these options is specified, then all the records in the secondary file are copied.

Ce3

Ce3 is a symbol made up of the key letter C and an expression e3, whose value must be positive number. If present, copying of records from the secondary file begins at the e3*th* character of each record.

Ce4

Ce4 is a symbol made up of the key letter C and an expression e4, whose value must be a positive number. If present, copying of records from the secondary file stops at the e4*th* character of each record.

If Ce3 but not Ce4 is given, copying begins at the e3*rd* character and extends to the end of each record.

If neither Ce3 nor Ce4 is given, each record copied is copied in its entirety.

KEY

This parameter indicates that eight numeric bytes from the secondary file record are to be used to create the key of the new principal file record. If the secondary input file is an ISAM file, the new record has the same key as the old record. If the secondary input file is an SAM file, the first eight bytes of the input record are used to form the ISAM key of the new record in the principal file.

ℓ

This parameter is a line number that does not exist in the principal file. New lines will be created beginning with this number. If ℓ is not given, then the lines are added to the end of the principal file.

In

This parameter may be used in conjunction with the ℓ parameter described above. *In* is a symbol made up of the key letter I and n, where n specifies the line number increment to be used to generate consecutive line numbers after ℓ. If this parameter is not specified, then the standard line increment is assumed.

Command Execution

The secondary file is opened for input and positioned to read the first record of the domain which may be one of the following:

1. e1*th* record, if Re1 is specified;

2. The record with Key k1, if Kk1 is specified and such a record exists in the file. If the record with key k1 does not exist, then the file will be positioned to be read from the next higher-valued key within the range.

3. First record of the file if neither of the above are given.

The specified range of each secondary file record is used to create a principal file record. If the ℓ and I parameters are used, the first line created in the principal file will have a number ℓ, and subsequent lines will have numbers in increments of n: ℓ+n, ℓ+2n, and so forth.

Thus, the ℓ and I parameters create a contiguous group of new records; whereas the KEY parameter can result in new records being scattered throughout the principal file.

Command Termination

Indicators:

#0	If at least one line is created, the command is said to be effective and #0 is set to 1. Otherwise, it is set to 0.
*	The * is pointed to the last line created.

Responses:

No review	No output to SYSOUT.
Partial review	List of line numbers created, their lengths, and the total count.
Full review	Every line created is printed out with its number and length and, finally, the total number of lines created.

Programming Notes

1. If a line in the domain has no content or it falls short of Ce1, then it is not used to create a new line. The next line in the domain is considered in that case.

2. This command is aborted if it is found before execution that ℓ already exists in the principal file.

3. If the KEY parameter is used, existing lines in the principal file may be replaced by newly created lines (if the old and new keys match). If the KEY parameter is not used, the command will be aborted if it attempts either to replace or to overstep an existing line in the file.

4. If the command is aborted during execution, all the lines that were already created remain in existence, * points to the last such line, and #0 is equal to 1.

5. In the parameter given syntactically as,

Re1 Re2

Kk1 Kk2

it is not necessary that k1 and k2 be existing lines in the secondary file. They serve only to define a range of lines in the secondary file to be copied.

6. e1 and e2 must be valid arithmetic expressions conforming to the rules set forth in Section 2 of this manual.

Example

Assume that there is a secondary file named SUBFL that has the following records:

```
CPDPOS          DC     A(0)
DFALSE          L      CREG4,CPDPOS
                L      CREG5,0(0,CREG4)
                SH     CREG4,=Y(4)
                ST     CREG4,CPDPOS
                DSKIP
```

and let the full review switch be on.

```
*GET SUBFL R4 R5 6 I2
        6.     <26>
         SH      CREG4,=Y(4)
        8.     <27>
         ST      CREG4,CPDPOS

NUMBER OF LINES CREATED:          <2>
```

HALT

The HALT command systematically terminates the File Editor session and returns control to the TSOS Command Language Processor.

Name	Operation	Operand
[Δ][.nameΔ]	{HALT / H}	[Ø]

No parameters are permitted.

Command Execution

This command will close either or both of the following files if they are open:

1. Principal file.
2. Procedure file.

Then it will release the File Editor from the user's virtual memory and return to the TSOS Command Language Processor. The TSOS Command Language prints a / on the terminal, indicating the end of the File Editor session. In the nonconversational mode, the TSOS Command Language processor reads the next command from SYSCMD.

Command Termination

Indicators: None.

Responses: In all review modes, the File Editor displays the name of any principal file closed, and also the name of any procedure file closed.

Example

Assume that the principal file being edited is named PAYROLL.VERSION3, and that no procedure file is in use.

```
*HALT
 CLOSED PRINCIPAL FILE PAYROLL.VERSION3
/
```

HELP

The HELP command displays on SYSOUT various messages designed to help the user of the File Editor.

Name	Operation	Operand	
[Δ][.nameΔ]	{HELP / HE}	[{ ⊅COMMAND / ⊅VERBLIST / ⊅PROCCALL / ⊅PROCDEFN / ⊅PROCNAME / ⊅ABBREV / ⊅any command-verb / ⊅any operand }]	[D]

COMMAND

A message is printed describing the general syntax of File Editor commands.

VERBLIST

Lists all File Editor verbs and procedure keywords

PROCCALL

The general form of a File Editor procedure call is shown.

PROCDEFN

A description of File Editor procedure definitions is printed.

PROCNAME

A description of procedure names is given.

ABBREV

The abbreviation scheme for File Editor commands is shown.

any command-verb

A message is given which lists all the operands that may be used with the given verb.

any operand

A short description of the given operand is printed.

If no operands are specified in the HELP command, an extended message describing the use of the HELP command itself is produced.

The various operands may be listed in any order in one HELP command; each operand must be preceded by at least one syntax delimiter, for example, a space or a comma.

Command Termination

Indicators: No Change

Responses: The messages described in the parameter descriptions. The setting of the review mode has no effect on the HELP command.

Example

The command

```
HELP      ALTER, A1, FIRST, PROCCALL
```

produces four messages, describing the ALTER command, the operands A1 and FIRST and procedure calls.

INPUT

The INPUT command creates new lines in the principal file, using textual data which it reads from SYSDTA. By this command, the user can create several lines in one terminal read operation, he can also create lines larger than the SYSDTA buffer.

Name	Operation	Operand
[Δ][.nameΔ]	{INPUT / I}	[Ᵽℓ][ⱣIn][ⱣX][Ⱪ]

ℓ

This parameter specifies the line number for the first line to be created. If specified, ℓ must not be an existing line number in the file. If it is not given, then the new lines are added to the end of the file.

In

In is made up of a key letter I and a variable n in one of the forms:

$d (d=0, 1, . . . 9)
$
*
integer of no more than 8 digits

specifying the line number increment to be used in generating consecutive line numbers after ℓ. If this parameter is not specified, then the standard line increment is used.

X

This symbol specifies that all input and output will be in two-character hexadecimal format. If X is not specified, the File Editor assumes that the input will be in character format.

Command Execution

The File Editor uses the text which it reads from SYSDTA to create lines in the following manner:

1. The first line consists of all characters up to the first paired carriage-return and line-feed (which may occur in either order); such a pair will be denoted here by CR/LF.

2. Each additional line consists of all characters after the last previous CR/LF and before the next CR/LF.

3. The CR/LF's themselves do not go into the file.

4. ETX's may be introduced at any point, without breaking the pattern of lines delimited by CR/LF. Thus, a line may be created in several pieces, each (except the last) ending with an ETX, and the last ending with CR/LF and (perhaps) an ETX.

5. ETX does not terminate the INPUT command, but merely allows the Editor to create lines from what has been already typed; after creating a group of lines, the File Editor returns to SYSDTA for more input.

6. To terminate the INPUT command, the user must type the character #END (or #E) either just after the File Editor has issued a new read to SYSDTA, or just after a CR/LF. The #END (#E) must be followed by an ETX, with perhaps one or more intervening spaces.

7. If, at the time #END is read, an incomplete line (not terminated by CR/LF) stands in the File Editor's INPUT buffer, that line will be made a line of the file.

8. If the X (hexadecimal) parameter was given, each two successive carriage-return/line-feed pairs must be separated by an even number of valid hexadecimal characters (0-9,A-F).

9. A carriage-return or line-feed standing alone will not be recognized and will be made a part of the line in which it stands.

10. If the beginning line-number ℓ was given and does not exist in the file, the first line created will have line-number ℓ; if ℓ was not given, the first line created will have line number $ + n if n was given; or $ + standard increment, if n was not given.

11. Under the full review mode, the list of lines created, lengths, and contents is output (if any lines have been created) each time the user types an ETX. Under partial review mode, the list of lines created and lengths is output either (1) at the end of the command, or (2) when the output buffer becomes full (every 56 lines).

Command Termination

Indicators:

#0	This indicator is set to 1 if at least one line is created; otherwise it is set to 0.
*	This indicator points to the last line created.

Responses:

No review	No output to SYSOUT.
Partial review	List of line numbers and their length followed by total number.
Full review	Line number, length, and content of each line created followed by total number.

1. If the user attempts to transmit a record longer than his task-buffer size, the File Editor will ignore that record, and ask him to send a shorter one.

The task-buffer obtained by the File Editor is 270 bytes long; however, if the user requested a larger buffer (up to 1024 bytes) in his LOGON command, this larger buffer is used by File Editor.

2. In order to put a line beginning with #END in this file, the user should prefix the line with a guard character (. in default). Any guard character found at the beginning of an input record will be stripped off by the File Editor. In order to create a line whose first character is the guard character, the user should prefix it with another guard character.

3. Tab characters may appear in the input, and will cause the File Editor to insert spaces up to the next tabulation position, if there is one. (See RESET Command, T operand.) If there are no more tabulation positions, the tab character is left as a valid character in the line created.

4. In the hexadecimal mode, each record must consist of one or more pairs of hexadecimal characters. Two adjacent tab characters are required to cause tabulation.

5. The Editor will abort the INPUT command and issue a diagnostic message if the user tries to replace or overstep an existing line. All lines previously created, of course, will remain in the file; * will point to the last line created.

6. As the Video-Data Terminal (RCA 70/752) has no line feed button, the single character carriage return is accepted by the INPUT command to stand for the pair CR,LF.

7. It is difficult to use the INPUT command in background mode, since key punches have neither CR nor LF. If desired, the EBCDIC equivalents of CR and LF (X'15' and X'25') may be multipunched to the right of the text in each card. The punch configurations are 11,9,5 (CR) and 0,9,5 (LF). (The TEXT command is much easier to use in background mode.)

8. If the input line is larger than the largest record permitted in the principal file, a truncated line is saved, and the command is aborted.

Example

```
DELETE 1,$
*INPUT 100,I5
*       PROCEDURE DIVISION.
        PARA1.
            OPEN INPUT FILEIN.
            READ FILEIN AT END GO TO FINI.
#END
*PRINT 1 $ N
      100.    <26>
       PROCEDURE DIVISION.
      105.    <13>
       PARA1.
      110.    <29>
           OPEN INPUT FILEIN.
      115.    <41>
           READ FILEIN AT END GO TO FINI.
*INPUT 113,I3
           PERFORM HSKPNG
           PERFORM PGHD.
#END
 KEY OF RECORD TO BE PROCESSED EXCEEDS KEY RANGE
  SPECIFIED     The input was terminated when the
                user tried to enter line 116, thus
                overstepping the existing line 115.

*PRINT 1 $ N
      100.    <26>
       PROCEDURE DIVISION.
      105.    <13>
        PARA1.
      110.    <29>
           OPEN INPUT FILEIN.
      113.    <25>
           PERFORM HSKPNG
      115.    <41>
           READ FILEIN AT END GO TO FINI.
```

JUMP

The JUMP command interrupts the sequence of execution of commands. The user may request an unconditional branch or may specify that the branch is to be executed only if a specified condition is met.

Name	Operation	Operand
[Δ][.nameΔ]	{JUMP / J}	[Ø[−].name2] { {Ø* / Ø$ / Ø$d} [Δ] { = / > / < / >= / <= / <> / Δ } [Δ] [a] [Ø] / {Ø# / Ø#d} [Δ] { = / > / < / >= / <= / <> / Δ } [Δ] [e] }

.name2

This parameter is optional; if it is specified, it indicates the name in the tag field of the command that is to be executed next if the condition specified in the operand field is met.

-.name2

The minus sign before .name2 indicates a jump reverse, which is permissible only inside a LOOP or a procedure.

If the .name2 field is not specified, the JUMP becomes ineffective (NOP).

$d
$
*

Identifies the symbolic line address counter whose value will be compared with the value of the address parameter a. If the result is true, the branch takes place.

#d
#

This symbol specifies the symbolic character counter whose value will be compared with the value of the given expression e. If the result is true, the branch takes place.

The comparison operators indicate the kind of comparison to be made between the preceding operands and those which follow. They are:

=	Jump if the two operands are equal.
>	Jump if the first operand exceeds the second.
<	Jump if the second exceeds the first.
>=	Jump if the first is greater than, or equal to the second.
<=	Jump if the first is less than or equal to the second.
<>	Jump if the two are unequal.
Δ	If one or more spaces (indicated in the command format by the delta symbol Δ) are present, the jump takes place if the two operands are equal. This feature is included for compatibility with earlier versions of File Editor.

Note: In paired symbols such as >= and <>, order is not significant. They could equally well be written => and ><.

a

a is a line address parameter and may be a line number, *, $, or $d reference, followed optionally by an arithmetic expression. If a is absent, the value of * is used.

e

e is an arithmetic expression that may contain decimal numbers, #, or #d references separated by the arithmetic operators + and -. If e is not present, the value 0 is used.

Command Execution

The execution of this command requires a comparison of the symbolic line address counter or symbolic character counter with the value specified by a (for line address counter) or e (for symbolic character counter); if the result is true, the next command to be executed is the one which has a name field corresponding to .name2. If the result is false, the next command in sequence is executed.

If there are no operands following .name2, the JUMP is considered to be an unconditional branch to .name2.

Command Termination

Indicators: No change.

Responses: None to SYSOUT.

Programming Notes

1. If a JUMP within a procedure definition attempts to jump outside that definition, the command is aborted, as is the Procedure mode.

2. Within a procedure definition, a forward JUMP or reverse JUMP is honored provided that .name2 can be found by searching in the specified direction.

3. If a JUMP command tries to reverse JUMP out of a LOOP sequence (see LOOP command), the LOOP will be terminated. The JUMP will be honored only if the LOOP sequence was read from a procedure definition; otherwise, it too will be terminated and a diagnostic message given.

4. A forward JUMP out of a LOOP sequence is permitted; the File Editor will attempt to satisfy the JUMP by reading from the present command source (SYSDTA or a procedure definition). No command read will be acted upon until the JUMP is satisfied.

5. In the operand field of the JUMP command, the operands may be separated by spaces, or not at all. However, no syntax delimiters other than spaces are acceptable. For example,

JUMP .MARY *=100 and
JUMP .MARY * = 100

are both acceptable, but

JUMP .MARY *,=,100

is not.

Example

```
*RESET R              (No review information required.)

*DELETE 1 $           (Delete all data from file.)

*INPUT 10,I10

*THE                  (A line-feed and carriage-return
                       was typed at the end of each line.)
QUICK

BROWN

FOX

JUMPS

OVER
```

```
THE

LAZY

DOG.

#END

*RESET  RF

 REVIEW MODE IS FULL

*SET * = 30

  * = 30.



*JUMP .DOG          (This is an unconditional jump and
                     would always be executed.)
*FIND 'CAT'

*PRINT *+5

*.DOG PRINT *

         30.       5

BROWN

*SET #2 = 10

  #2 = 10

*JUMP .DOG #2=3+7   (This jump would be executed
                     because #2=10.)

*.CAT ALTER *,*,'MVC','MVI'

*PRINT *+5

*.DOG PRINT 10,20

*THE
 QUICK
```

```
*SET #2=345

   #2=345



*JUMP .DOG #2=3+7
*.CAT PRINT 40

  FOX

*.DOG PRINT  *+1

  JUMPS
```

(This jump would not be executed because #2 is not equal to 10.)

LOOP

The LOOP command executes a sequence of commands repeatedly until a specified termination condition is met.

Name	Operation	Operand			
[Δ][.nameΔ]	{LOOP / L}	⫫.name2	{⫫a5 / ⫫Xe}	[⫫a6]	[⫫]

.name2

.name2 is the name appearing in the name field of the statement that is to terminate the Loop sequence.

a5 and a6

After executing the command statement named .name2, the Loop sequence is reentered if * is such that a5 ⩽ * ⩾ a6. Otherwise, the loop is terminated.

If neither a5 nor a6 is specified and Xe is not specified, the current value of * at LOOP command recognition is taken for a5 and a6. If a6 is not specified, then a6 equals a5.

Xe

Xe is a parameter consisting of the key-letter X and an expression e, whose value must be positive. If this parameter is given, the loop sequence will be executed e times.

a5 D a6
Xe

These two options are mutually exclusive. If the Xe form is not explicitly stated, then the File Editor assumes the a5, a6 form.

Command Execution

1. Commands are read from the present command source (SYSDTA or a called procedure), up to and including the one that contains .name2 in its name field. This series of commands, called the Loop sequence, is saved in virtual memory; these commands will be executed until some condition is met.

2. Execution begins with the first command of the sequence, and normally terminates with the last. If the Xe form of the command was given, the entire sequence will be executed e times (or until an error occurs). If the a5 [⫫ a6] form was given, the sequence will be executed, and at the end, * will be compared to a5 and a6. If * is not outside the range a5 to a6, inclusive, the sequence will be executed again, and so on. If * is outside this range, the Loop execution will terminate, and the File Editor will read a new command from the present command source.

3. If a forward JUMP cannot be satisfied within the LOOP sequence, the LOOP terminates, but the JUMP is honored, and must be satisfied from the present command source before any new command will be executed.

4. If a reverse JUMP cannot be satisfied within the LOOP sequence, the LOOP is terminated. The JUMP is honored only if the LOOP sequence was read from a procedure definition; otherwise, it too will be terminated, and a diagnostic messages given.

5. At LOOP termination, the entire sequence (but not the memory acquired to hold it) is discarded, and may be reinvoked only by being read in again from some command source.

6. The total length of the code generated by the LOOP sequence commands should not exceed 4050 bytes, which is more than enough for almost all users. However, an experienced user who is entering an exceptionally large LOOP sequence may wish to see whether he is in danger of running out of space; he can do so by escaping from the File Editor (by depressing the ESC key) and using IDA to scan the page where LOOP coding is stored. File Editor acquires page $(63)_{10}$ $(3F_{16})$ to store the code of a Loop sequence, so the following IDA command

/DISPLAY L'3FF00':L'3FFFF'

lets the user look at the last 256 bytes of that page to try to determine whether he is approaching his limit.

It should be emphasized, however, that only in very exceptional cases does the user need to worry about this.

Command Termination

Indicators: Remain set at whatever values were last given to them within the Loop sequence.

Responses: None from the LOOP command itself.

Programming Notes

1. Although a LOOP command may occur within a procedure definition, a procedure call is not permitted within a LOOP sequence.

2. The commands in the LOOP sequence should not occupy a total of more than 4050 bytes.

Example A

This loop removes trailing spaces from all lines of the principal file.

```
*FIND   1,$                    (Set to 1st line)
*LOOP   .END,0,$               (establishes loop)
*SET    #3=#                   (point #3 to end of current line)
*FIND   *,C#3,C#3,"Δ"          (present line ends with space?)
```

Note: Δ denotes space character; at terminal, user depresses space bar.

```
*JUMP  .END,#0=0               (no, go to .END)
*.A SET #3=#3-1                (decrement column pointer)
*FIND *,C#3,C#3,"Δ"            (present line ends with space?)
*JUMP -.A,#0=1                 (yes, go back to .A)
*SET #3=#3+1                   (set col. pointer to 1st space)
*DELETE *,C#3                  (delete group of trailing spaces)
*.END SET*=*+1                 (get next line, go to head of loop)
```

Example B

Assume that the principal file contains the following records:

```
     100.     <24>
          CLI   AAA,X'00'
     200.     <19>
          BH    DINT
     300.     <24>
BEGINX    MVI   AAA,X'05'
     400.     <24>
          CLI   BBB,X'06'
     500.     <19>
          BO    DINT
     600.     <21>
          B     DSKIP
     700.     <24>
BEGINY    MVI   CCC,X'11'
     800.     <24>
          MVI   BBB,X'17'
     900.     <24>
          MVI   CCC,X'ST'
```

```
     1000.      <19>
          B       DINT
     1100.      <3>
AAA
     1200.      <3>
BBB
     1300.      <3>
CCC
     1400.      <4>
VALA
     1500.      <4>
VALB
     1600.      <4>
VALC
     1700.      <3>
ABC
```

Assume also that the following procedure, named LOOPR, is available for the user to execute. (It is not necessary that the reader understand File Editor Procedures at this stage, the example is designed merely to demonstrate the execution of the LOOP command).

```
BEGIN LOOPR
S $7=1100
S $8=1400
LOOP .ENDLP1,X3
ALTER 100 950 C16 L$7 L$8 ALL
JUMP .ENDLP #0=1
NOTE NONE FOUND
.ENDLP S $7=$7+1
S $8=$8+1
.ENDLP1 NOTE END OF LOOP EXEC'N
END LOOPR
```

X3 specifies that the loop is to be performed 3 times

The results of executing the procedure LOOPR with full review appear below:

```
*LOOPR
 $7 = 1100.

 $8 = 1400.

       100.     <25>
            CLI    VALA,X'00'
       300.     <25>
```

```
BEGINX    MVI   VALA,X'05'
 NUMBER OF LINES AFFECTED:   2
$7 = 1200.

$8 = 1500.

.ENDLP1 NOTE END OF LOOP EXEC'N
     400.     <25>
         CLI    VALB,X'06'
     800.     <25>
         MVI    VALB,X'17'
 NUMBER OF LINES AFFECTED:    2
$7 = 1300.

$8 = 1600.

.ENDLP1 NOTE END OF LOOP EXEC'N
     700.     <25>
BEGINY   MVI    VALC,X'11'
     900.     <25>
         MVI    VALC,X'ST'
 NUMBER OF LINES AFFECTED:    2
$7 = 1400.

$8 = 1700.

.ENDLP1 NOTE END OF LOOP EXEC'N
```

(End of first execution of loop)

(End of second execution of loop)

(End of 3rd execution of loop)

MOVE

The MOVE command copies a range of records from one area of the principal file to another area of the same file; that is, it creates a new domain inside the same principal file.

Name	Operation	Operand
[Δ][.nameΔ]	{MOVE / M}	[{{ℙ[a1 ℙ[a2]] / ℙa1 ℙa1 / ℙa1} [Ce1[ℙCe2]]} [ℙℓ]][ℙIn]

a1 and a2

This pair of line address parameters indicates the range of lines of the principal file to be processed. If a2 is not specified, the value of a1 is taken for a2 also. If neither a1 nor a2 is specified, * is used for both.

Ce1 and Ce2

This pair of line content parameters defines a contiguous portion of every line of the range. The quadruple a1, a2, Ce1, Ce2 defines the domain of the command.

If Ce2 is not specified, processing extends from Ce1 to the end of the line. If neither Ce1 nor Ce2 is specified, the entire line is copied.

ℓ

This parameter is a line number that does not exist in the principal file. New lines are created beginning with this number. If ℓ is not given, then the lines are added to the end of the principal file. In order for this parameter to be recognized, either a1 and a2 must be present, or Ce1 must be present. Otherwise, what the user intends as ℓ will be interpreted by the File Editor as either a1 or a2.

In

In is a symbol made up of key letter I and a variable n, specifying the line number increment to be used to calculate succeeding line numbers after ℓ. If this parameter is not specified, then the standard line increment is assumed.

Command Execution

The MOVE command copies all lines from a specified domain of the principal file, creating a series of new lines in the same sequence. The first line is ℓ (or, in default, $+n) and subsequent lines have line number ℓ+n, ℓ+2n,. . .

Command Termination

Indicators:

#0	If at least one new line is created, then the command was effective and #0 is set to 1. Otherwise, it is set to 0.
*	The * points to the last line created.

Responses:

No review	No output to SYSOUT.
Partial review	List of line numbers created, their lengths, and the total count.
Full review	The text of every line created is printed out along with its number and length and finally the total number of lines created.

Programming Notes

1. If a line in the principal file has no content in the domain, then it is not used to create a new line. The next line in the domain is then processed.

2. This command is aborted if it is found before execution that ℓ already exists in the principal file.

3. The MOVE command will not allow a user to inadvertently destroy valid data in the principal file. Thus, if a key about to be created already exists in the file, the command will be aborted, and a diagnostic message printed. The same is true if, in creating new keys, the MOVE command tries to overstep an existing line.

4. If the command is aborted during execution, then all the lines that were already created are preserved and * points to the last such line. Also, #0 is equal to 1.

Example

```
*DELETE 1 $
*INPUT  5 I5
 *CPDPOS     DC     A(0)
DFALSE  L    CREG4,CPDPOS
        L    CREG5,0(0,CREG4)
*       SH   CREG4,=Y(4)
        ST   CREG4,CPDPOS
        B    DSKIP
#END

*RESET  RF
  REVIEW MODE IS FULL
*MOVE 16,26,6,I2
         6.     26
        SH       CREG4,=Y(4)
         8.     27
        ST       CREG4,CPDPOS
NUMBER OF LINES AFFECTED:          2

*PRINT 1 $ N
         5.    <19>
CPDPOS     DC    A(0)
         6.    <26>
           SH    CREG4,=Y(4)
         8.    <27>
           ST    CREG4,CPDPOS
        10.    <27>
DFALSE     L     CREG4,CPDPOS
        15.    <31>
           L     CREG5,0(0,CREG4)
        20.    <26>
           SH    CREG4,=Y(4)
        25.    <27>
           ST    CREG4,CPDPOS
        30.    <20>
           B     DSKIP
*
```

NOTE

The NOTE command displays on SYSOUT the entire contents of the statement in which it appears. It thus demonstrates that this statement has been executed.

Name	Operation	Operand
[Δ][.nameΔ]	{NOTE / N}	[Ⓓ any characters][Ⓓ]

The operand field is optional and, if present, is written on SYSOUT.

Command Execution

The entire command statement is displayed on SYSOUT.

Command Termination

Indicators: not applicable.

Responses: not applicable.

Programming Notes

1. Even if the listing option is specified, the NOTE command is listed only once.

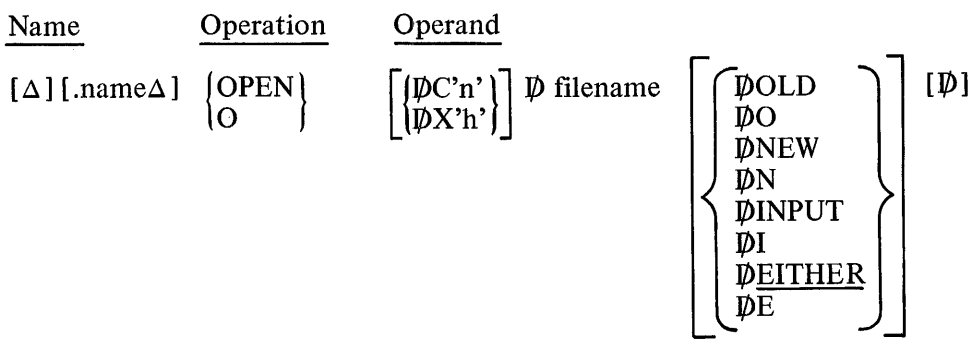
2. The NOTE command is designed especially for use inside LOOP, PROCEDURE, or nonconversational sessions to indicate that the command processing has reached certain specified points, or to indicate the actual flow of execution in the above situations.

Example

```
*.ENDLP NOTE REACHED END OF LOOP
```

OPEN

The OPEN command closes the principal file currently being edited, if any, and opens the named file as the principal file.



C'n'
X'h'

Either of these parameters supplies the WRITE password required to access the next principal file. Absence of this parameter indicates that no password is required to access the named file.

filename

This parameter identifies the next principal file to be opened.

OLD or O

Specifies that the file to be opened is an old file and is to be opened in INOUT mode, that is, File Editor can read and write it. An old file is one which has been written to at least once.

NEW or N

The file to be opened is a new file and is to be opened in OUTIN mode, that is, File Editor can write it, then read it. A new file is one which has never been written into, whether or not it has been cataloged or allocated.

INPUT or I

The file to be opened is an old one and is to be opened in the INPUT mode, that is, File Editor can read it but not write it.

EITHER or E

The file is to be opened either OLD or NEW, depending on whether such a file already exists or not.

The default value for this parameter is EITHER.

1. If there is a principal file already open, it will be closed, and a message printed to that effect.

2. The File Editor determines the status of the specified file.

If the file is new, and the user specifies the parameters OLD (O) or INPUT (I), he is notified of his error. The parameters NEW and EITHER are both acceptable. If the file has not been cataloged, or has been cataloged but not allocated, File Editor catalogs it (if necessary) and allocates space for it. File Editor specifies a primary and secondary space allocation for a new file of 3 half-pages of 2048 bytes each (2 tracks on a 70/564, 1 track on a 70/590).

If the file is old, the parameter NEW (or N) is an error and the user is so notified. OLD, INPUT, and EITHER are all acceptable. After opening an old file, File Editor tests its characteristics and, if they do not conform to its own requirements, closes it immediately and sends an error message to the user. The characteristics of a file acceptable to the File Editor are:

File-type = ISAM

Record-type = F or V

Key-length - 8

If F-type records, record size $\leqslant$ 2048 bytes.

If V-type records, block size = 2048 bytes.

3. The File Editor applies a standard set of characteristics to any new file it is asked to open. These characteristics are:

File-type - ISAM

Record-format - V-type, with no print-control character

Key-length - 8 bytes

Key-position - 5 (immediately after four-byte record-length field)

Block-size - 2048 bytes

Retention-period - 0 days

Padding-factor - 20%

Initial space allocation - 3 PAM pages (that is, 2 tracks on a 70/564, 1 track on a 70/590)

Automatic secondary allocation - 3 PAM pages

No read or write passwords

Write access

Not sharable

Direct-access, public volume

4. If the file is successfully opened, File Editor sends the user a message containing the filename, the record-type, the status (OLD or NEW) of the file, and, if INPUT was specified, a reminder that the file cannot be written to.

Command Termination

Indicators:

1. If the file is empty, *,$, all the $d pointers and # are set to zero. All the #d pointers are set to 1.

2. If the file is not empty, * and all the $d pointers are set to the line number of the first line of the file and # is set to the length of that line. $ is set to the line number of the last line of the file and all the #d's are set to 1.

Responses:

In any review mode, a message notifies the user of the name of any file closed and the name of the file opened.

Programming Notes

1. If the named file cannot be opened for any reason, any previous principal file will still be closed; in this case an error message follows the informational message saying that a file was closed.

2. To create a new file with special characteristics, such as a password or shareability, the user may issue the DMS commands/FILE or /CATALOG before opening it with File Editor. The /FILE command must contain the parameter LINKNAME=PRINFILE. See the Data Management System Reference Manual.

Example A

Assume that the name of the current principal file is PAYROLL.VERSION6 and the user wants to close it and create a new file to be named PAYROLL.VERSION7.

```
  User:  OPEN PAYROLL.VERSION7,NEW
System:  CLOSED PRINCIPAL FILE PAYROLL.VERSION6
         OPENED PAYROLL.VERSION7 AS NEW
         V-TYPE FILE.
```

Example B

To create a new file with fixed-length records of 256 bytes each, an initial and secondary allocation of 10 PAM pages, and a write password:

```
User:       /FILE   NEWFILE,LINK=PRINFILE,SPACE=(10,10),
                    RECFORM=F,RECSIZE=256,KEYPOS=1

          /CATALOG  NEWFILE,STATE=UPDATE,WRPASS=C'5053'

         /PASSWORD  C'5053'

            /EXEC   (EDIT)

TSOS:       %P001 - DLL  V-XX

Editor:     VERSION XXXX OF FILE EDITOR READY

User:       *OPEN NEWFILE, NEW

Editor:     OPENED NEWFILE AS NEW F-TYPE FILE
```

Note that the password could have been given in the File Editor OPEN command rather than in the Data Management PASSWORD command.

PRINT

The PRINT command writes a domain of the principal file to the SYSOUT device.

Name	Operation	Operand
[Δ][.nameΔ]	{PRINT / P}	[ⱿaI[ⱿaI]][ⱿCeI[ⱿCe2]][ⱿN][ⱿX][Ɀ]

a1 and a2

This pair of line address parameters indicates the range of lines of the principal file to be processed.

If a2 is not specified, only the line indicated by a1 is printed. If neither a1 nor n2 is specified, the line indicated by the current line pointer * is printed.

Ce1 and Ce2

This pair of line content parameters defines a contiguous portion of every line in the range. The quadruple a1, a2, Ce1, Ce2 defines the domain of the command.

If Ce2 is omitted, each line is printed from Ce1 to the end. If both Ce1 and Ce2 are omitted, the entire line is printed.

N

If N is specified, the line number and length are printed prior to the content of each line. If N is not specified, the line number and associated length are not printed.

X

If X is specified, the line is printed in two-character, hexadecimal format. If X is not specified, the line is printed in normal character format.

Command Execution

This command prints out the specified lines on SYSOUT. If N is given, the line number and the number of characters to be printed out of that line are printed before the actual text.

After the whole domain has been printed, the File Editor reads the next command. If the next command is a null command, that is, an ETX only, the next existing line, if any, is printed out according to Ce1, Ce2, N, and X as given in the original command. Reading SYSDTA and printing the next line continues until a nonnull command is read or the end of the file is reached. If the end of the principal file is reached, the PRINT command is aborted, an error message is printed, and the next command is read.

Command Termination

Indicators: * The value of * is set to the line address of the line last printed.

Responses: Not applicable.

Programming Notes

1. If full or partial review is on, the line numbers and lengths are printed, regardless of whether N was specified or not.

2. Null lines are ignored by PRINT; that is, the File Editor passes them by without informing the user of their presence.

3. Lines in the range which are not null but are still shorter than Ce2 are treated by this command like null lines.

4. Note that the length printed out with the line number is not the total length of the line but is the number of characters actually printed.

Example

```
*DELETE 1 $              (clear out existing content of file)

*INPUT 100 I100          (and insert new text)

*HOW NOW BROWN COW

TO BE OR NOT TO BE
ALL   IS      LOST
#END

*INPUT 400 I100

 DATA
BASE
IS
VERY SMALL AS YET
#END

*PRINT 100,300,C14

  COW
 TO BE
 LOST
*PRINT 400,700,C6,C11

  SMALL
*
```

QUALIFY

The QUALIFY command opens a procedure file and makes all valid procedures in that file available for execution.

Name	Operation	Operand	
[Δ][.nameΔ]	{QUALIFY / Q}	[{ⱣC'n' / ⱣX'h'}]	Ᵽ filename [Ᵽ]

C'n'
X'h'

C'n' or X'h' is the read password that is associated with the named procedure file. If this is not given, no password should be required to read the named file.

filename

This parameter supplies the name of an ISAM file that contains one or more File Editor procedures.

Command Execution

During a File Editor session, only one procedure file may be active at a time. The QUALIFY command causes the designated file to be opened for input (read only). The QUALIFY command acquires user memory, reads the procedure file, and creates a directory of procedures. The user is then notified of the number of valid procedures found and the name of the last existing procedure. If another procedure file had previously been QUALIFIED, it is closed, its directory released, and the new procedure file is then activated as described above.

Command Termination

Indicators: Not applicable.

Responses: See Command Execution.

Programming Notes

1. The user must QUALIFY a procedure file before he may call any of the procedures which it contains.

2. Once the procedure file has been opened, it remains open until either:

   a. A new QUALIFY command is given.

   b. A CLOSE command is given for that file.

   c. The File Editing session is terminated.

3. If a user qualifies a file containing no valid procedure definitions (that is, ordered pairs of valid BEGIN and END statements), he is given a diagnostic message, and the file will be closed.

4. If a procedure file contains some valid definitions along with some invalid ones, the valid ones will be made available for execution, and the invalid ones will cause diagnostic messages to be printed.

5. Only one page (4096 bytes) of user's memory is obtained to hold the procedure directory; because of this limitation, only the first 170 valid procedure definitions in a procedure file will be made available for execution. If a procedure file contains more than this number, the user will be notified by a diagnostic message.

Example

The command:

```
QUALIFY C'VVP1',    PROC
```

causes the opening of a file called PROC whose associated password is VVP1.

RESEQUENCE

The RESEQUENCE command allows the user to renumber the lines of the principal file.

Name	Operation	Operand
[Δ][.nameΔ]	{RESEQUENCE / RESEQ}	[␣a][␣In][␣]

a

The new line number (record key) to be assigned to the first line in the file. If this parameter is omitted, the value of the parameter In is used.

In

This parameter consists of the key-letter I and an increment n in one of the following forms:

$d where d = 0,1. . .9
$
*

or a decimal number which conforms to line number requirements. If In is omitted, the current value of the standard line number increment is used. The In parameter is used as an increment to calculate subsequent line numbers after the first.

Command Execution

The File Editor copies the entire contents of the principal file to a temporary secondary file; as each record is copied, it is erased from the principal file. When the principal file is empty, the records in the secondary file are copied back into the principal file, with new record keys assigned. After recopying is completed, the temporary secondary file is erased.

Command Termination

Indicators:

#0	Set to 1 if at least one record is created; otherwise set to 0.
*	Points to the last line created.

Responses:

No review	No output to SYSOUT.
Partial review	List of new line numbers and lengths, followed by total number of lines resequenced.
Full review	New line numbers, lengths and contents, followed by total number of lines resequenced.

Programming Notes

File Editor names the temporary secondary file:

FILE.EDITOR.SAM.RESEQUENCE.FILEd

where d is a decimal digit 0-9. If it happens that all ten such filenames are already in use when the command is given, File Editor notifies the user and terminates the command.

Example

Given a file with lines:

```
100.  HOW NOW BROWN COW
110.  TO BE OR NOT TO BE
120.  ALL IS LOST
130.  DATA
140.  BASE
150.  IS
160.  VERY SMALL AS YET
```

Suppose the full review switch is on.

```
 User:     RESEQUENCE 100,I100
Editor:    100.  <17>  HOW NOW BROWN COW
           200.  <18>  TO BE OR NOT TO BE
           300.  <11>  ALL IS NOT LOST
           400.   <4>  DATA
           500.   <4>  BASE
           600.   <2>  IS
           700.  <17>  VERY SMALL AS YET
```

RESET

The RESET command allows the user to change those File Editor variables which define the session attributes.

Name	Operation	Operand
[Δ][.nameΔ]	{RESET / R}	[Ⓓλ][Ⓓλ[Ⓓλ[. . .]]] [Ⓓ]

where:

λ = {Ⓓ+s / Ⓓ-s / Ⓓ} {Gs / G} {In / I} {LY / LN / L} {Pd / P} {RF / RP / RN / R} {TA / TC / TF / T} [+n-m . . .]

There are seven key-letter variables that may be modified by the user. The following rules govern their use:

1. Each parameter specified must be preceded by at least one syntax delimiter.

2. Any of the seven parameters may be used more than once in one Reset command. (See examples below.)

3. The parameters are entirely nonpositional and are recognized only by their key letter.

4. If the key letter but no value is given, the File Editor default value is reinstated.

The seven parameters are discussed below:

1. Text Delimiter:
   D+s
   D-s
   D

where s is a string of characters consisting of regular characters or hexadecimal characters. In the case D+s, all characters in the string are added to the current list of delimiters. D-s deletes all characters in the string s from the current list of delimiters.

D causes the list of File Editor defaults to be reinstated. These are ( ) space ; : . , ! ' ' ' ? = + - * / . It is considered an error if one tries to reinstate an existing delimiter or delete a nonexistent delimiter.

Guard Character:

Gs
G

The File-Editor-supplied guard character is the period (.). The period is reestablished by issuing the second form above (G). The first form (Gs), where s is a string of one character, resets the current guard character to a new character. The user is cautioned against using the / or # characters as the guard character.

3. Standard Line Number Increment:

In
I

The default increment is 00000100. This value is reinstated by issuing the second form above (I). In the first form, In , n consists of either

a. One to eight decimal digits.

b. A reference to one of the pointers *, $, $0, . . . , $9.

4. Listing:

LY
LN
L

LY (List Yes) means that each command is to be displayed on SYSOUT before it is executed. LN (List No) means that commands are not to be displayed. This facility is designed for the nonconversational user and for the user who is trying to debug a procedure or a LOOP command. L has the same effect as LN.

5. Decimal Pointer:

Pd
P

This parameter establishes the apparent decimal pointer for keys in the principal file. The d may be a single decimal digit with a value of 0 through 8. The second form (P) reestablishes the File Editor default, which is a decimal point following the eighth digit. (This form has the same effect as P0.)

6. Review:

RF
RP
RN
R

This parameter changes the amount of review information desired in response to the execution of a command. RF means full review; RP means partial review; and RN and R mean no review. Specific details of the resulting output are given with each command.

7. Tabs:

TA
TC +n-m . . .
TF
T

The 10 available tabs can be set as follows:

TA sets Assembler tabs at columns 10, 16, 72;

TC sets COBOL tabs at columns 8, 12, 73;

TF sets FORTRAN tabs at columns 7, 73; and

T eliminates all tabs.

TA+n-m . . . resets Assembler tabs, then adds tab n and deletes tab m; similarly, TC+n-m and TF+n-m.

T+n-m leaves established tabs as they are and then adds n and deletes m.

Command Execution

See parameter description above.

Command Termination

Indicators: See description above.

Responses:

No review None.

Partial review None.

Full review All values of variables set by this command are printed on SYSOUT.

Programming Notes

See example.

Example

```
*

*RESET RF
 REVIEW MODE IS FULL
*RESET D
 FILE EDITOR DELIMITERS
  .(+!*);-/,?:'="
*RESET D-''''
 FILE EDITOR DELIMITERS
  .(+!*);-/,?:="
*RESET D+'A'
 FILE EDITOR DELIMITERS
  .(+!*);-/,?:="A
*RESET G'='
 GUARD CHARACTER = '='
*RESET G
 GUARD CHARACTER = '.'
*RESET I$4
 LINE INCREMENT = 0.
*RESET I2
 LINE INCREMENT = 2.
*RESET P3 P4
 DECIMAL POINTER = 3
 DECIMAL POINTER = 4
*RESET R,TA,D+'BCD',RF T+2,R
 REVIEW MODE IS FULL
  FILE EDITOR TABS
 2       10      16      72
*VERIFY R,T,D
 REVIEW MODE IS OFF
  FILE EDITOR TABS
 2       10      16      72
 FILE EDITOR DELIMITERS
  .(+!*);-/,?:="ABCD

*RESET
*V
 FILE EDITOR DELIMITERS
  .(+!*);-/,?:'="
```

```
GUARD CHARACTER = '.'
$0 = 0.  $1 = 0.  $2 = 0.  $3 = 0.  $4 = 0.
     $5 = 0.  $6 = 0.  $7 = 0.  $8 = 0.
  $9 = 0.  $ = 0.  % = 0.
#0 = 1  #1 = 1  #2 = 1  #3 = 1  #4 = 1  #5 = 1
#6 = 1  #7 = 1  #8 = 1  #9 = 1  # = 0
LINE INCREMENT = 100.
DECIMAL POINTER = 0
REVIEW MODE IS OFF
 FILE EDITOR TABS ARE NOT DEFINED.
```

SEARCH

The SEARCH command finds and displays all instances of a given string in a particular range of the principal file.

Name	Operation	Operand
[Δ][.nameΔ]	{SEARCH / SEA}	[⍁a1[⍁a2]][⍁Ce1[⍁Ce2]][⍁s][⍁N][⍁X][⍁]

a1 and a2

This pair of line address parameters defines the range of lines in the principal file to be scanned. If a2 is absent, only a1 is scanned. If a1 and a2 are both absent, the line indicated by the current line pointer, *, is scanned.

Ce1 and Ce2

This pair of line content parameters defines a contiguous portion of every line in the range. If Ce2 is omitted, File Editor scans every line from the *e1st* column to the end; if both Ce1 and Ce2 are absent, the entire line is scanned.

The quadruple a1, a2, Ce1 and Ce2 defines the domain of the command.

s

Specifies the string for which File Editor is to search the domain. If s is omitted, then any lines which intercept both Ce1 and Ce2 are found.

N

If this key-letter parameter is specified, every line printed out is prefaced by its line number and length.

X

If this key-letter parameter is specified, all lines printed are given in hexadecimal format.

Command Execution

If s is given, the domain a1, a2, Ce1, and Ce2 is scanned for the string s. Every line where s occurs is printed on SYSOUT.

If s is not given and Ce1 and Ce2 are both specified, all lines in the range a1, a2 that have at least e2 characters are printed.

If s is not given, and Ce1 is specified but not Ce2, all lines in the range a1, a2 that have at least e1 characters are printed.

If none of the parameters s, Ce1 and Ce2 are specified, File Editor displays all lines in the range a1, a2 on SYSOUT.

If N is specified in the Search command or the current review mode is full or partial, every line displayed is prefixed by its line number and length.

If File Editor cannot find any line that meets all criteria, it displays a message to that effect on SYSOUT.

Command Termination

Indicators:

#0	Set to 1 if one or more lines are displayed; otherwise, set to 0.
*	Points to the last line in the domain.

Programming Notes

1. If s is null and neither Ce1 nor Ce2 is specified, File Editor looks for null lines. Each time it finds one, it displays the message <NULL LINE>; if the parameter N was specified, or review mode is full or partial, the message is prefaced by the line number of the null line and a length of zero.

2. If s is null and Ce1 is specified (with or without Ce2) File Editor displays an error message and terminates the Search command.

Example

Assume that the review mode is full and that the principal file contains the following lines:

```
100.   SYMB1     DC     A(TIMB1)
200.   SYMB2     DC     A(TIMB2)
300.   SYMB3     DC     A(TIMB3)
400.   MMB1      MVC    MB1,SYMB1
500.             MVC    SYMB2,MB1
```

The command

```
SEARCH    100,$,'SYMB1'
```

produces the following results

```
100         <23>
SYMB1        DC       A(TIMB1)
400         <24>
MMB1         MVC      MB1,SYMB1
```

The command

```
SEARCH 100,$,C10,'SYMB1',N
```

produces

```
400       <24>

MMB1       MVC MB1,SYMB1
```

even though the current review mode is N (none).

SET

The SET command may be used to assign a value to a symbolic line pointer, the current line pointer or a symbolic character counter.

Name	Operation	Operand	
[Δ][.nameΔ]	{SET / S}	[{␣$d [␣a] / ␣* [␣a] / ␣#d [␣e]}]	[␣]

$d␣a

Resets the value of a symbolic line pointer $d. ␣a, which must be present, should be a decimal digit with a value of 0 through 9. a may be any line address parameter; if it is omitted, the value of * is taken.

*␣a

Sets the current line pointer to a new value; at the same time, # will automatically be set to the length of a. If a is omitted, the File Editor will not change any values.

#d␣e

Resets the value of a symbolic character counter #d. d, which must be present, must be a decimal digit with a value of 0 through 9. e may be any arithmetic expression consisting of decimal numbers or symbolic counters joined by plus or minus signs; if e is omitted, the value 1 is used.

If the user issues a Set command with no parameters whatsoever, * and the 10 $d's are set to the address of the first line in the file; the 10 #d's are set to 1 and # to the length of the first line of the file.

Command Execution

See above.

Command Termination

Indicators: Not changed.

Responses:

No review	None.
Partial review	None.
Full review	The values of all variables set by this instruction.

Programming Note

None.

Example

```
*SET
 $0 =.    $1 = 0.    $2 = 0.    $3 = 0.    $4 = 0.

        $5 = 0.    $6 = 0.   $7 = 0.    $8 = 0.   $9 = 0.

     $ = 0.    * = 0.

#0 = 1    #1 = 1    #2 = 1    #3 = 1    #4 = 1    #5 = 1

#6 = 1    #7 = 1    #8 = 1    #9 = 1    # = 0

*RESET RF
 REVIEW MODE IS FULL
*SET $9=1050
 $9 = 1050.

*SET #9=600
 #9 = 600
*SET #4=20
 #4 = 20
*SET #5=#9-#4
 #5 = 580

*SET * = 53
   * = 53
```

TEXT

The TEXT command permits the user to scan his file and make changes or create new lines.

Name	Operation	Operand
[Δ] [.nameΔ]	{TEXT / T}	[ⱣDa] [ⱣDIn] [ⱣDX] [ⱣD]

a

If this parameter is specified, processing starts at line number a. If this parameter is not specified, processing begins at the end of the file.

In

In is made up of the key letter I and a variable n of the form:

$d or

* or

integer of no more than 8 digits.

This parameter specifies the increment value to be used during the execution of this command and is called the temporary increment. If this parameter is omitted, then the standard increment controlled by the RESET command is used.

The increment (either the specified temporary or default standard) determines the next line to be processed in the following manner: If the current line is k, then the next line is either k + n or the next existing line of the file, whichever is first.

X

This key-letter parameter indicates that input and output will be in hexadecimal format for this command. If this parameter is omitted, the File Editor assumes character format input and output.

Command Execution

Processing begins with line a, its length and its contents being printed if it exists. If a does not exist, then the number a is printed (and length zero). A read is then issued to the terminal.

The user may then take one of two specific actions:

1. Send from SYSDTA a record containing several characters not beginning with #. This will have the effect of either creating line *, if line * does not already exist, or replacing the contents, regardless of length, of line * with the record. Note, if the user wishes to input a line with # as the first character, then he must precede it with the guard character. The guard character is stripped off, and the remaining contents are stored.

2. Issue one of the following subcommands:

a. #a,text

This command will replace or create line a using the specified text. When line a has been edited, control will return to the current line. a is a line number consisting of one to eight digits; there must be no space between the #, the line number and the comma.

b. #

This command leaves the current line undisturbed and processing continues with the next line, as described above.

c. #PRINT

The symbol # followed by a syntactically correct File Editor Print command will display the requested lines. Control returns to the current line after these lines have been printed. Note: The null command option of the PRINT command is not allowed in #PRINT.

d. #DELETE

The symbol # followed by a syntactically correct File Editor DELETE command will delete the requested domain of the principal file and then return control to the current line.

e. #END [Δ] or #E[Δ]

This subcommand terminates the TEXT command.

Command Termination

Indicators: * points to the last line created or displayed.

Responses:

No review — Line numbers, length, and original contents of lines edited will be displayed on SYSOUT.

Partial review — In addition to the above, line numbers and new lengths of edited lines will be displayed, and also the total number of lines edited.

Full review — In addition to all the above, the new contents of each edited line will be displayed.

Programming Notes

In creating new lines, the TEXT command modifies the records which are input by SYSDTA as follows:

1. It deletes all line-feed and carriage-return characters from the input.

2. It recognizes tab characters, and inserts an appropriate number of spaces.

3. It deletes the first character of the record if this is the guard character.

4. If the user responds with the null command (ETX only) then that line becomes null.

5. The TEXT command reads data from the terminal using the tandem write-read (WRTRD) macro. Therefore, after the File Editor types the current line number and length and existing text, if any, the user may begin typing data without waiting for an asterisk (*) to be printed. This is illustrated in the TEXT command example.

Example

```
*OPEN ROVER, NEW
 OPENED ROVER AS NEW V-TYPE FILE
*RESET I200
*TEXT  100
       100. <0>  |  SUBROUTINE ALPHA (BETA)
       300  <0>  |  COMMON GAMMA (3,5),
                 |  DELTA (10), EPSILON
       500  <0>  |  PARAM=BETA
       700  <0>  |  #350, COMMON THETA
       700  <0>  |  10 FORMAT (5X,I8)
       900  <0>  |  #70A, (FORMAT 5X,I4)

%D T00C ILLEGAL CHARACTER IN LINE ADDRESS.

       900  <0>  |  #700,10 FORMAT (5X,I4)
       900  <0>  |  DO 25 I=1,3
      1100  <0>  |  #DELETE 350
      1100  <0>  |  #
      1300  <0>  |  DO25 J=1,5
      1500  <0>  |  #950, GAMMA (1,1)=PARAM
      1500  <0>  |  GAMMA (I+1,J)=GAMMA(I,J)
                 |  *PARAM
      1700  <0>  |  #E
*
```

TYPED BY FILE EDITOR

TYPED BY USER

FILE EDITOR ERROR MESSAGE

TYPED BY FILE EDITOR

TYPED BY USER

The file would then appear as follows:

```
  100    <22>    SUBROUTINE ALPHA (BETA)
  300    <35>    COMMON GAMMA (3,5), DELTA (10), EPSILON
  500    <10>    PARAM=BETA
  700    <15>    10 FORMAT 5X,I4
  900    <11>    DO25 I=1,3
  950    <16>    GAMMA (1,1)=PARAM
 1300    <11>    DO25 J=1,5
 1500    <29>    GAMMA (I+1,J)=GAMMA(I,J) *PARAM
```

UPDATE

The UPDATE command inserts a string of characters before or after a specific character in an existing line.

Name	Operation	Operand
[Δ] [.nameΔ]	{UPDATE / U}	[ɃaI [Ƀa2]] [ɃCe] [{ɃSUFFIX / ɃS / ɃPREFIX / ɃP}] Ƀs [Ƀ]

a1 and a2

This pair of line address parameters defines the range of lines of the principal file to be processed.

If a2 is omitted, then a1 is the only line edited. If neither a1 nor a2 is specified, the line indicated by the current line pointer * is edited.

Ce

This parameter consists of the key letter C and an expression designating a column number within the record. If specified, it determines where within the record the string s is to be placed. If not specified, either the beginning or the end of the line is assumed, depending upon whether PREFIX or SUFFIX is specified.

PREFIX
or
P

Explicit use of this parameter indicates that the string s is to precede the *eth* character of the line.

SUFFIX
or
S

This parameter indicates that the string s is to follow the *eth* character of the line. If neither S or P is explicitly specified, then SUFFIX is assumed.

s

A valid string must be given.

Command Execution

1. The string given by s is inserted into each line in the range at a point determined by the parameters Ce and PREFIX or SUFFIX.

2. If Ce is not given and SUFFIX is intended, the string s is added to the right of each line.

3. If Ce is not given and PREFIX is specified, the string given by s is inserted at the beginning of each line.

Command Termination

Indicators:

#0	Set to 1 if any line in the domain is updated; otherwise, set to 0.
*	The value of * is set to the address of the last line scanned in the domain.

Responses:

No review	None.
Partial review	List of line numbers and new lengths of lines affected and total count.
Full review	Line numbers, new lengths, new contents of lines affected, and total count.

Programming Note

If a line becomes larger than the maximum allowed for the principal file, a truncated line is saved and the command aborted.

Example

```
*RESET     RF

 REVIEW MODE IS FULL
*PRINT 10,20,N

        10.     <27>
           MVC    LAB1,LABTEMP
        15.     <26>
           AH     REG3,Y(523)
        20.     <27>
           STH    REG3,LABTEMP


*UPDATE 15,C20,SUFFIX,'='

        15.     <27>
           AH     REG3,=Y(523)
  NUMBER OF LINES AFFECTED:      <1>
*
```

VERIFY

The VERIFY command displays on SYSOUT various File Editor pointers and session attribute switches.

This command is not positionally oriented, and so it departs from the standard syntax format.

Name	Operation	Operand
[Δ][.nameΔ]	{VERIFY / V}	[{Ⱳβ [Ⱳβ[Ⱳβ [. . .]]] / ⱲX}] [Ⱳ]

where: $\beta = \{ * / \$[d] \} \mid \#[d] \mid D[X] \mid G[X] \mid I \mid P \mid R \mid T$

There are 11 keyword (letter, symbol) variables that may be displayed for the user. The following rules govern their use:

1. If two or more parameters are used, they must be separated from one another by delimiters -- Ⱳ.

2. Any one of the 11 parameters may be used more than once in one Verify command.

3. The parameters are entirely nonpositional and are recognized only by the key letter.

Command Execution

1. * displays the current value of *.

2. $d displays the current value of the specific symbolic line pointer; d must be a single decimal digit from 0 through 9.

3. $ displays $0,. . ., $9, $,*.

4. #d displays the current value of the desired symbolic character counter; d must be a single decimal digit from 0 through 9.

5. # displays the current values of #0, #1, . . ., #9, and #.

6. Delimiters D[X]

D displays the list of delimiters in regular character format; DX displays the list in two-character hexadecimal format.

7. Guard Character G[X]

G displays the guard character as a regular character; GX displays the character in two-character hexadecimal format.

8. Increment I

I displays the current value of the standard line number increment.

9. Decimal Pointer P

P displays the number of digits to the right of the assumed decimal pointer for principal file line keys. This will be a decimal number from 0 through 8.

10. Review R

R displays the degree of review currently being given (full, partial, or none).

11. Tabs T

T displays the various current tab settings.

12. If no parameters are given, then all variables are displayed as regular characters.

13. There is a special form of the Verify command that displays all the above variables in regular character format except for the guard and delimiter characters, which it displays in two-character hexadecimal format. The syntax of this command is as follows:

Name	Operation	Operand
[Δ][.nameΔ]	{VERIFY / V}	X [ⱣⱣ]
		ⱣⱣ

X as a parameter can be used only as shown in this syntax; if it is used otherwise, an appropriate error message is printed.

Command Termination

There is no change in indicator values, and the review mode has no effect.

Example

```
*SET *=53
 * = 53.

*SET $5=207
 $5 = 207.

*SET #7=23
 #7 = 23
*VERIFY *,$5,#7
  * = 53.

 $5 = 207.
```

```
 #7 = 23
*VERIFY
 FILE EDITOR DELIMITERS
  .(+!*);-/,?:'="
 GUARD CHARACTER = ','
 $0 = 0.   $1 = 0.   $2 = 0.   $3 = 0.   $4 = 0.
   $5 = 207.   $6 = 0.   $7 = 23. $8 = 0. $9 = 1050.
     $ = 0.   * = 53.
#0 = 1  #1 = 1  #2 = 1   #3 = 1   #4 = 20  #5 = 580

#6 = 1    #7 = 23   #9 = 600  # = 0
LINE INCREMENT = 100.
DECIMAL POINTER = 0
REVIEW MODE IS FULL
 FILE EDITOR TABS ARE NOT DEFINED.

*VERIFY X
 GUARD CHARACTER = '4B'
 FILE EDITOR DELIMITERS IN HEXADECIMAL
 404B4D4E5A5C5D5E60616B6F7A7D7E7F

 $0 = 0. $1 = 0.  $2 = 0.  $3 = 0.  $4 = 0.
 $5 = 207.   $6 = 0.   $7 = 0.   $8 = 0.$9 = 1050.
    $ = 0.     * = 53.

#0 = 1   #1 = 1   #2 = 1   #3 = 1   #4 = 20

#5 = 580   #6 = 1   #7 = 23   #8 = 1  #9 = 600

# = 0
LINE INCREMENT = 100
DECIMAL POINTER = 0
REVIEW MODE IS FULL
 FILE EDITOR TABS ARE NOT DEFINED.
```

WRITE

The WRITE command copies a domain of the principal file into a sequential (SAM) secondary file.

Name	Operation	Operand
[Δ][.nameΔ]	{WRITE / W}	[Ⓓ {ⒹC'n' / ⒹX'h'}] Ⓓ filename [Ⓓa1[Ⓓa2]] [ⒹCe1 [ⒹCe2]] {ⒹKEY / ⒹK} [{ⒹOLD / ⒹO / ⒹNEW / ⒹN}] [Ⓓ]

C'n'
X'h'

This parameter specifies the Write password required to open the secondary file. If omitted, no password should be required to write to this file. (See note on passwords, Section 3.)

filename

This parameter, which must be given, specifies the filename of the secondary file to be written to.

a1 and a2

This pair of line address parameters defines the range of lines of the principal file to be processed.

If a2 is omitted, then a1 is the only line written. If both a1 and a2 are omitted, the line indicated by the current-line pointer, *, is written to the secondary file.

Ce1 and Ce2

This pair of line content parameters defines a contiguous portion of every line in the range. The quadruple a1, a2, Ce1, Ce2 defines the domain of the command.

If Ce2 is omitted, copying extends from Ce1 to the end of each line. If both Ce1 and Ce2 are omitted, complete lines are copied.

KEY
K

If this parameter is specified, the File Editor constructs a secondary file record thus: the first eight bytes consist of the ISAM key of principal file record. The remainder of the secondary file record is formed from the specified portion of the principal file record. (The secondary file created by WRITE will be a SAM file, and so has no keys of its own.) Thus, principal-file keys, as well as the contents of records, may be saved in the secondary file.

NEW
N

This parameter indicates that the user wants to erase the present contents, if any, of the secondary file and write the designated lines of the principal file, starting at the beginning of the secondary file.

OLD
O

This parameter indicates that the user wants to append the designated lines of the principal file onto the present contents of the secondary file, leaving the present contents unchanged.

Command Execution

1. If NEW (or N) is specified and the named file does not exist, it will be cataloged, allocated three PAM pages with an automatic extension of three PAM pages, and opened as a SAM file in OUTPUT mode. (This means that the output-record-pointer will be positioned to the beginning of the file.)

2. If NEW is specified and the file exists, it will be opened as a SAM file in the OUTPUT mode. If the named file has been cataloged but not allocated, it will be allocated as above, then opened for OUTPUT.

3. If OLD (or O) is specified and the file exists, it will be opened in the Extend mode (that is, the output-record-pointer will be positioned to the end of the file).

4. If OLD is specified and the file does not exist (or has not been allocated), the File Editor will abort the command and issue a diagnostic message.

5. After opening the secondary file as described above, the command copies the specified domain of the principal file into the secondary file, then closes the secondary file.

Command Termination

If at least one record is successfully written out, the command is said to be effective and #0 is set to 1. Otherwise, it is set to 0.

* Points to the last line written out.

Responses:

No review	None.
Partial review	Line number and size of each record written; also the total count.
Full review	Content, number, and size of each line written out and also the total count.

Programming Notes

1. The command passes over any lines that are shorter than e1 and considers the next consecutive line inside the domain.

2. The user may, if he wishes, supply all information about a new file through the TSOS commands CATALOG and FILE before issuing a WRITE command to the File Editor. The FCB linkname for the secondary file is SECFILE.

Example

```
*DELETE    1    $

*RESET     RF

*INPUT     5,I5

*JB'S OPINIONS
SHALL REMAIN
FOREVER CLOAKED
IN MYSTERY
#END
         5.     <13>
 JB'S OPINIONS
        10.     <02>
 SHALL REMAIN
        15.     <05>
 FOREVER CLOAKED
        20.     <00>
 IN MYSTERY
  NUMBER OF LINES AFFECTED:      <4>
*WRITE SECFL,10,20,NEW
 SHALL REMAIN
 FOREVER CLOAKED
 IN MYSTERY
  NUMBER OF LINES AFFECTED:      <3>
*
```

A procedure in the File Editor system is a group of commands taken from the File Editor vocabulary that together perform a more complicated operation than is possible with a single verb of the language. Each procedure can be thought of as a macro within the File Editor structure. Furthermore, procedures are not sets of commands that are limited to one particular form; rather, they are generalized commands carrying parametric and symbolic forms into which the user may inject specific values so as to cause a particular procedure to perform a particular task.

THE PROCEDURE DEFINITION

The procedure definition is the outline of the procedure. It contains one or more extended File Editor commands, together with identifying information at the beginning and end of the definition. Thus, a procedure definition begins with the keyword BEGIN, or its initial B, and ends with the keyword END, or its initial E; the line containing BEGIN must also contain the name of the procedure. The name of the procedure must begin with an alphabetic character, followed by one to seven other characters which may be alphabetic or numeric. For example, a procedure for searching out a particular word might begin with:

```
BEGIN SRCHWRD
```

and end with:

```
END
```

or

```
END SRCHWRD.
```

Between the opening and closing lines of a procedure definition, there may be one or more File Editor commands. Each such command may carry with it symbolic parameters. In general, these parameters are shown in the form :n, where n can be any decimal integer. Typical symbolic parameters, therefore, are:

```
:1          :34          :5          :14
```

Thus, one finds that typical statements in a procedure definition look like the following examples:

```
SET * = :1
FIND *,:2,':3'
VERIFY *
```

The following is a typical procedure definition consisting of eight statements:

```
BEGIN SCRHWRD

SET * = :1

LOOP .LABEL :1,:2

FIND *,:2,':3'

JUMP .LABEL #0=0

VERIFY *

.LABEL SET *=*+1

END SCHRWRD
```

Such a procedure definition can be built up interactively by the user and stored for future reference.

Procedure Files

File Editor procedure definitions are stored in a File Editor procedure file and called forth at execution time. The user creates a File Editor procedure file just as he would create any other file, by opening it as a new file and inserting text (in this case, procedure definitions) into it by means of the TEXT or INPUT commands.

The characteristics of a valid File Editor procedure file are:

1. File-type = ISAM

2. Record-type = Variable, with no print-control characters.

3. Key-length = 8

4. Key-position = 5 (immediately after 4-byte record-length field)

5. Block-size = 2048

After the user has inserted procedure definitions in his procedure file, and closed the file, he can then make it available as a source of procedure definitions by issuing a QUALIFY command.

When a procedure file is qualified, that is, made available for procedure calls during a File Editor session, only one page of virtual memory is allocated for the directory of procedures. One page accommodates 170 procedures, and if a procedure file contains more than this number, the user is notified that the directory space has been exhausted. However, the first 170 procedures are accessible.

Because a user may have his procedure definitions residing in different files, he must indicate to the File Editor which file he wishes the procedure calls to refer to. He does this by issuing the QUALIFY command. For example, the command

QUALIFY PROCFILA

opens PROCFILA and creates a directory of the procedures in that file. Any procedure residing in PROCFILA may now be invoked by a procedure call.

A procedure call consists of the name of the procedure followed by the desired values of the symbolic parameters. The name of the procedure is placed in the operation field, and the various parameters are listed in the operand field, P1, P2, P3,. . .,Pn.

Name	Operation	Operand
[Δ][.nameΔ]	procedure-name Δ	P1,P2,P3, . . . , Pn

The parameters must be separated by commas. If there are no parameters in the call, the Δ following the procedure name may be omitted.

A parameter may not exceed 256 characters and must meet the following requirements:

1. The beginning of a parameter in the operand field is indicated by the first nonspace character in the operand field or the first nonspace character following the comma that indicated the end of the previous parameter.

2. The end of a parameter is indicated by the ETX or by the next comma which is not embedded in a quoted string.

3. If a quoted string appears in the parameter, the rules given in Section 2 for string parameters must be observed.

When the procedure call is issued, the accompanying parameter values are inserted into the procedure definition. This process is called expansion. To illustrate the expansion of a procedure definition, consider the following example:

The procedure named BECKON is defined as follows:

```
BEGIN BECKON

FIND :2

PRINT :1

ALTER :1,:2,':3',':4'

END BECKON
```

This procedure is a general statement of a program that finds a specific line in the principal file, prints out a line from the principal file, and substitutes a new string for the first occurrence of another string within a range of lines.

In order to call this procedure, the user might write

BECKON *,*+5,40,50

The expansion of the procedure definition follows the rule that the order of the parameter values in the procedure call corresponds to the numerical value in the definition. Thus, the value :2, in the FIND statement, looks to the second position in the parameter lineup in the call statement; the value given there is *+5. Similarly, the :1 following PRINT becomes *. The fully expanded procedure would then be as follows:

FIND *+5

PRINT *

ALTER *,*+5,'40','50'

Note: The *+5 means the fifth line after the *.

Each procedure definition line is expanded and then executed.

A procedure call is not permitted within a Loop sequence. For example,

```
LOOP .BOTTOM X3
.
.
.
BECKON 100,200,300
.
.
.
.BOTTOM
```

is illegal. The Loop sequence will be executed up to BECKON and then will halt with a diagnostic message.

VARIATIONS OF PARAMETER REPRESENTATIONS

Symbolic Parameter: 0

The name field of the procedure call, in addition to its standard use, is also considered a parameter and is associated with the special symbolic parameter, :0. Except for this special numbering convention, the item in the name field will be treated during the expansion in exactly the same way as the parameters given in the operand field of the call. For example, assume the procedure call:

.TAG BECKON 100,200,'ABLE','BAKER'

The procedure definition statement: ALTER :1,:2,:3,':0' will be expanded as ALTER 100,200,'ABLE','.TAG'.

Null Parameters and Null Symbolic Parameters

If a parameter is omitted from the operand field of a procedure call, such a parameter becomes a null parameter. The value of a parameter is limited to the extent between commas; every parameter except the last should be followed by a comma.

The following is an example of a call in which the third parameter is null:

```
BECKON 100,5000,,LABEL
```

If this same call had been written as:

```
BECKON 700
```

the last three parameters would be considered null.

When a symbolic parameter in a procedure definition statement corresponds to a null parameter in a procedure call, it is said to be a null symbolic parameter. During expansion, a null character value will replace the null symbolic parameter. The effect is the same as if the symbolic parameter did not appear in the statement.

For example, given the call,

```
BECKON 100,500,IT
```

and the line,

```
FIND :1,:2,":3"
```

the expanded statement would be:

```
FIND 100,500,"IT".
```

But, if the call were

```
BECKON 200,,IT
```

the expanded statement would be:

```
FIND 200,,"IT".
```

Concatenation

A symbolic parameter in a procedure definition statement may be immediately preceded or followed by other characters or another symbolic parameter. When the command is expanded, the characters that correspond to the symbolic parameter are combined with the other characters or the value associated with the other symbolic parameter. For example, in

```
FIND 100,500,':3S'
```

let the value associated with :3 be TAPE; the expanded command becomes:

```
FIND 100,500,'TAPES'.
```

There are two chaining situations that require special conventions to avoid ambiguity.

Concatenation of a Symbolic Parameter with a Suffixed Decimal Digit

If one wishes, for example, to concatenate :1 with the suffixed decimal digit 7, the problem is to avoid confusing this combination with :17 which identifies symbolic parameter 17. To avoid this ambiguity, the sign @ (at the rate of) is designated as a separator. The @ is placed between the symbolic parameter number and the numerical value to be concatenated with the specific value. For example, the procedure definition statement

FIND 100,500, ':3@2'

with the value TAPE for the third parameter, would be expanded as:

FIND 100,500, 'TAPE2'

Concatenation of the Colon Character with a Suffixed Decimal Digit

Because a symbolic parameter is represented by a colon followed by one or more decimal digits, there must be a special notation to indicate when the same sequence of characters is to be interpreted literally and not as a symbolic parameter. The convention adopted is to use two consecutive colons when a literal interpretation is wanted. Thus, :1 means symbolic parameter 1, but ::1 means a colon followed by a decimal 1. The ::1 will be replaced by :1 during expansion.

The following rules govern the use of colons:

1. If the sequence colon-followed-by-a-digit is intended, then the colon must be preceded by another colon. (Illustrated by example below.)

2. If the sequence colon-followed-by-any-other-character is intended, it does not matter whether there are one or two colons (see examples b and d below).

3. Two consecutive colons in a procedure definition statement are always replaced by a single colon during expansion, as in example b and c below. They will not be considered when identifying the beginning of a symbolic parameter in a procedure definition statement.

The following examples illustrate the application of the colon rules. Assuming that the procedure call assigned a value LABEL to :1, the model statements on the left are shown on the right in their expanded form.

Model Statement	Generated Command
a. FIND 100,500,':1'	FIND 100,500,'LABEL'
b. FIND 100,500,'::SYMBOL'	FIND 100,500,':SYMBOL'
c. FIND 100,500,'::1'	FIND 100,500,':1'
d. FIND 100,500,':SYMBOL'	FIND 100,500,':SYMBOL'

Inner Calls

A procedure can contain as a part of its procedure definition another procedure call. The call appears as a procedure definition statement and is termed an inner call. The call that invoked the procedure containing the inner call is considered to be the outer call. All symbolic parameters that appear in an inner call are replaced by the values assigned to them by the outer call, just as in the expansion of any other line of the procedure definition.

Nesting of procedures in this manner may go to any depth. Recursive procedure calls are also permitted.

Example 1

```
*OPEN PTFILE,NEW

OPENED PTFILE AS NEW V-TYPE FILE.

*INPUT 10
*BEGIN CALL0
FIND :2
PRINT :1
ALTER :1,:2,':3',':4'
END CALL0
#END
*PRINT 1 $
BEGIN CALL0
FIND :2
PRINT :1
ALTER :1,:2,':3',':4'
END CALL0
*HALT
/LOGOFF BUT
%B003 LOGOFF AT 1332 ON 09/29/69, FOR TSN 2108.
%B014 CPU TIME USED: 0018.5120 SECONDS.
%B001 PLEASE LOGON.
/LOGON USERID
%B002 LOGON ACCEPTED AT 1333 ON 09/29/69. TSN 2127
   ASSIGNED
/EXEC EDIT
%L001 PROGRAM LOADING
 VERS. 0005 OF FILE EDITOR RDADY
*OPEN EK1
 OPENED EK1 AS OLD V-TYPE FILE.
*QUALIFY PTFILE
 NUMBER OF ENTRIES CREATED: 1
NAME OF LAST ENTRY: CALL0
```

```
*D 1 $
*INPUT 10,[10
*AAA
BBB
CCC
DDD
EEE
FFF
GGG
HHH
III
JJJ
KKK
LLL
#END
*SET *=10
*RESET LY
*CALL0 *,*+5,GGG,XXX
 CALL0 *,*+5,GGG,XXX
 FIND *+5
 PRINT *
 FFF
 ALTER *,*+5,'GGG','XXX'
*PRINT 1 $
 PRINT 1 $
 AAA
 BBB
 CCC
 DDD
 EEE
 FFF
 XXX
 HHH
 III
 JJJ
 KKK
 LLL
```

(To ensure that command statements will be displayed on terminal before they are executed.)

Example 2

```
*/EXEC (EDIT)
%L001 DYNAMIC LOADER INVOKED
 VERS. 0009 OF FILE EDITOR READY
*RESET G'='
*OPEN PTFL NEW
 OPENED PTFL AS NEW V-TYPE FILE.
*INPUT I10
*BEGIN SRCHWRD
SET *=:1
LOOP .LABEL :1 :2
FIND * :2 ":3"
JUMP .LABEL #0=0
VERIFY *
=.LABEL SET *=*+1
END SRCHWRD
#END
*HALT
/LOGOFF BUT
%B003 LOGOFF AT 1524 ON 10/10/69, FOR TSN 3363.
%B014 CPU TIME USED: 0016.0643 SECONDS
%B001 PLEASE LOGON.
/LOGON USERID
%B002 LOGON ACCEPTED AT 1524 ON 10/10/69, TSN 3375
     ASSIGNED
/EXEC (EDIT)
%L001 DYNAMIC LOADER INVOKED
 VERS. 0009 OF FILE EDITOR READY
*OPEN PRINC
OPENED PRINC AS OLD V-TYPE FILE.
*DELETE 1 $
*INPUT 100 I50
*THIS FILE
CONTAINS TEST
DATA FOR THE
PURPOSE OF
MAKING A TEST
OF THE F.E. PROCEDURE
FACILITY. TEST FILE ONLY
#END
*PRINT 1 $ N
      100.      <9>
 THIS FILE
      150      <13>
 CONTAINS TEST
      200      <12>
```

```
 DATA FOR THE
      250      <10>
 PURPOSE OF
      300      <13>
MAKING A TEST
     350      <21>
OF THE F.E. PROCEDURE
     400      <25>
FACILITY. TEST FILE ONLY.
*QUALIFY PTFL
 NUMBER OF ENTRIES CREATED: 1
 NAME OF LAST ENTRY: SRCHWRD
*RESET LY
*SRCHWRD  100,500,TEST
 SRCHWRD  100,500,TEST
 SET *=100
 LOOP .LABEL 100 500
 FIND * 500 "TEST"
 JUMP .LABEL #0=0
 VERIFY *
  * = 150.

 .LABEL SET *=*+1
 FIND * 500 "TEST"
 JUMP .LABEL #0=0
 VERIFY *
 * = 300.

 .LABEL SET *=*+1
 FIND * 500 "TEST"
 JUMP .LABEL #0=0
 VERIFY *
  * = 400.

 .LABEL SET *=*+1
 END OF FILE OCCURRED READING THE PRINCIPAL FILE
*HALT
 HALT

 PROCEDURE FILE CLOSED
/LOGOFF
%B003 LOGOFF AT 1532 ON 10/10/69, FOR TSN 3375
%B014 CPU TIME USED: 0008.5525 SECONDS.
```

Title TSOS File Editor Reference Manual

Document No. DJ-003-2-00

Date May 1971

RCA Computer Systems

Your comments and suggestions will help us to furnish publications that are more useful to you.

Is this publication:

Complete in its coverage?

Logically organized?

Technically accurate?

Easy to understand?

Other comments (Use additional page if necessary).

Name _____ Street or Box No. _____

Job Title _____ City _____

Company _____ State _____ Zip _____

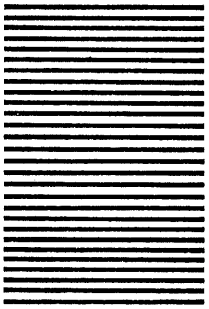
Cut Along Line

Fold

FIRST CLASS
PERMIT NO. 16
CAMDEN, N. J.

BUSINESS REPLY MAIL — no postage necessary if mailed in the United States

Postage will be paid by addressee



RCA | COMPUTER SYSTEMS
DATA PROCESSING DIVISION
CAMDEN, N. J. 08101

ATTN: Marketing Publications
Bldg. 204-2, Cherry Hill

Fold

Publications Purchase Order

RCA|Computer Systems Division
Camden, N. J. 08101

Order Number

Item No.	Quantity Ordered	Ordering Number	Description or Title of Material	Complete if Publications Are To Be Purchased	
				Unit Price	Totals
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Ship To:

Bill To: (Complete if Publications Are To Be Purchased)

Customer P.O. #

Sub Total	
Indicate Applicable Sales Tax	
Total Cost	

Check Appropriate Block:

Bill My Company | |

Remittance Enclosed | |

If Publications are to be purchased forward this Form and any enclosures to:

RCA | Computer Systems Division
Reproduction Services
Camden, N. J. 08101

Publications purchased will be furnished subject to all terms and conditions stated on the reverse side of this Form.

Ship via

Date Required

Authorized Signature

Date

All other requests: Forward this form to the nearest RCA District Office.

Terms And Conditions
Applicable To The Sale of RCA Publications and Forms

RCA

PRICES

All prices are subject to change or withdrawal without notice and all shipments will be billed at prices in effect on date of shipment. Unless otherwise specified or required by law, all prices will be billed exclusive of state and local sales and similar taxes, and such taxes will appear as additional items on invoices.

TRANSPORTATION

All shipments will be f.o.b. destination.

On shipments where Purchaser requests transportation involving expenses beyond those involved on transportation normally selected by RCA, the Purchaser will be responsible for payment of such extra costs.

RCA reserves the right to ship from any location subject to the foregoing transportation terms.

DELIVERIES

It is the desire of RCA to meet requested delivery schedules. However, RCA shall not incur any liability due to any delay or failure to deliver for any reason. Any delivery indication furnished by RCA only represents the best estimate of the time required to make shipment. The delivery of part of any order shall not obligate RCA to make further deliveries, and RCA reserves the right to decline servicing any order in whole or in part.

Of necessity, inventories and current production must be allocated in such a manner as to comply with applicable Government regulations. In the absence of such regulations, RCA reserves the right to allocate inventories and current production when, in its opinion, such allocation is necessary.

TERMS OF PAYMENT

Invoices shall be rendered at time of shipment and shall be payable net 30 days from date of shipment.

Partial shipments will be invoiced as made, and payments therefor are subject to the above terms.

GENERAL

In no event shall RCA be liable for indirect, consequential or special damages.

Information furnished by RCA is believed to be accurate and reliable. However, no responsibility is assumed by RCA for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of RCA.

This Agreement shall be governed by the laws of the State of New York and constitutes the entire Agreement between the parties with respect to the subject matter hereof. It shall prevail regardless of any variation in the terms and conditions of any other submitted by the Purchaser.

INDEX