SPERRY

PUBLICATIONS RELEASE

Series 1100

Text Editor
Level 16R1C

Reference

UP-8723.2

This Library Memo announces the release and availability of *Series 1100 Text Editor, Level 16R1C, Reference,* UP-8723.2. This is a Standard Library Item (SLI).

This manual corresponds to the Text Editor (ED Processor) level 16R1C. This release is a stability update and as such does not contain any new features. Refer to the software release documentation that accompanies the Text Editor level 16R1C for a list of the modifications made.

The Text Editor (ED Processor) allows you to create and edit a symbolic file or element. Using the Text Editor you can insert, delete, and replace both ASCII and Fieldata text. This manual serves as a reference for the users of the Text Editor level 16R1C. It describes the editing commands used to maintain symbolic files and elements.

The Text Editor Level 16R1C software described in this manual is a proprietary product of Sperry. Therefore, authorization to receive the software package must be established by contractual agreement prior to transmittal.

Additional copies of the manual may be requisitioned through your Sperry representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists AC, BZ, CZ, MZ	Mailing Lists 8, 9, 9U, 30, 31U, 37, 37U, 38, 62, 63, 63U, 64, 64U, 81, 81U, 83, 83U, 89, 89U, MBR, MBSU, MCRU (80 pages)	Library Memo for UP-8723.2
		RELEASE DATE: April 1984

Series 1100

Text Editor

Level 16R1C

Reference

SPERRY

A

UP-8723.2

Page Status Summary

Section	Pages	Update
Cover/Disclaimer		
PSS	1	
Preface	1	
Contents	1 – 3	
Section 1	1 – 4	
Section 2	1 – 38	
Section 3	1 – 13	
Section 4	1 – 7	
Section 5	1 – 6	
Index	1 – 3	
User Comment Sheet		

Section	Pages	Update

A vertical bar (|) in the outer margin of the updated pages indicates technical changes.

Preface

The Series 1100 Text Editor (ED Processor) allows you to create and edit a symbolic file or element. Using the Text Editor, you can insert, delete, and replace both ASCII and Fieldata text.

This manual is to serve as a reference for the users of the Text Editor. It describes the editing commands used to create, update, or delete symbolic files or elements. It also describes LOOP operations and usage considerations.

An additional manual that may be useful is the *Series 1100 Executive System, EXEC, Programmer Reference* , UP-4144. (Use the version that corresponds to the EXEC software level in use at your site.) You may order this through your Sperry representative.

Contents

1. Introduction

1.1. General

The Series 1100 Text Editor (ED Processor) allows you to create or edit a symbolic file or element conversationally. Using the Text Editor, you can insert, delete, and replace both ASCII and Fieldata text.

Using the text editor you can:

- View the text in a variety of forms: send output to a terminal, an onsite printer, or to an onsite card punch; request line numbers for each line of text; and print text in quick mode (i.e., omitting all nonsignificant blanks).
- Alter or delete existing lines, and insert new lines (using tabs if desired).
- Locate strings of character in the text.
- Copy or move lines of text from one section of your data base to another.
- Add sections of text from other files or elements to the existing text.
- Create new files and elements from existing text.
- Use an internal pointer to keep track of line numbers.
- Read ASCII, Fieldata, or mixed ASCII-Fieldata text, and edit or print ASCII or Fieldata text.
- Use indentation (tabs) for formatting lines.
- Write programs to perform complex repetitive tasks.
- Name sequences of commands as macro commands and call them by name.

1.2. Text Editor Call Statement Format (@ED)

Purpose

To call the Text Editor (ED Processor) and specify its input, output, and operation modes.

All parameters of the @ED control statement are optional.

Format

```
@ED,options   name-1,name-2
```

Parameters

options	see Table 1-1.
names	specifies input or output files or elements (see Table 1-1).

Description

Table 1-1 lists the available options, the input/output files as specified by *name-1/name-2*, and functions. Unless otherwise specified, *name-1* and *name-2* may be either files or elements.

If you omit *name-1*, the Text Editor will assume the name of an element in TPF$. If you omit *name-1* and do not specify any options of the set C, I, R, U, the Text Editor will assume the C option.

```
@label:ED,options   name-1,name-2
```

The Text Editor operates in two modes: input and edit. In input mode all lines entered are directly inserted into the text. In edit mode, you can use various commands to modify existing text. To change modes, enter a blank line (press the TRANSMIT key). Most editing commands refer to a particular part of the text. You can position the cursor directly, or you can use the LOCATE, FIND, and CHANGE commands to position the cursor to the line that you wish to edit.

If you do not specify the P or Q options, the output file will be the same character set as the input file. If the input file is mixed, the type of the output file will be the same as the type of the label image of the input file. This means that print files will usually be Fieldata, so the Q option will be required for ASCII editing. If you are creating a new element using the I option implicitly or explicitly, the mode will be Fieldata unless you specify the Q option.

The Text Editor will allow you to print files without destroying the line spacing. Any new lines that you insert have a default spacing of one. You can print an edited file (@SYM) as a normal print file with one exception: 0 becomes 077.

If your output (*name-1* for I option, *name-2* if no option or if used with U option) is an element that duplicates the name of an already existing element in the output program file, you will receive a warning message. A decision on whether to replace the existing element is up to you.

If the file the Text Editor returns to you exceeds the maximum size, the Text Editor will reassign the file 10 percent larger (see 2.2.15). It will then display the message:

```
OUTPUT FILE OVERFLOW
```

Table 1-1. Text Editor Control Statement (@ED) Options

Option Character	*name-1*	*name-2*	Description
Not I, R, or U	Input	Output	Input is taken from *name-1* and the resultant text is placed in *name-2*.
Not I, R, or U	Input	None	R option assumed for symbolic element. U option assumed for data file.
A	Input	Output	Attempt auto recovery - see 2.2.4.
B	Input	Output	Batch mode when using a demand terminal - the Text Editor will not solicit input from user (see 2.2.37). This may also be achieved by OFF TREAD.
C	Input or Output	Ignored	Enter input mode (I option) if element does not exist. Otherwise, assume U option.
D	Input	Output	Demand mode when using a batch terminal - output listing of the Text Editor run will contain solicitation messages.
E	Input	Output	Set EOF mode on at start of edit (see 2.2.37). This may also be achieved by ON EOF.

I	Output	Ignored	Initial insertion of symbolic input from the runstream which causes the Text Editor to enter the input mode. The images following the @ED control statement are inserted into the file or element named in *name-1*. The I option takes precedence over the R or U option.
			
			If the I option is specified or if two fields are specified, then the Text Editor checks the output file (i.e., field 1 if I option; field 2 if two fields are specified).
			
			If the output is a data file and you specified an element in that file, then the Text Editor terminates in error after printing the error message:
			
			`FILE NOT PROGRAM FILE FORMAT`
			
			If the output file is a program file and you do not specify an element in that file, then the Text Editor prints a warning message:
			
			`WARNING: ON EXIT OUTPUT FILE WILL BECOME A DATA FILE`

K	Input	Output	Inverts the ASCII-Fieldata indicator bit (bit 0 of the first 050 SDF control word).
L	Input	Output	Print all lines following the @ED control statement. The lines printed are indented and preceded by four asterisks. This option may be altered by ON/OFF LISTINP.
N	Input	Output	Suppresses printing of changed, found, or relocated lines. This option serves the same purpose as the ON BRIEF command and may be altered by ON/OFF BRIEF.

Table 1-1. Text Editor Control Statement (@ED) Options (continued)

Option Character	*name-1*	*name-2*	Description
P	Input	Output	Output file will be Fieldata. This mode can also be set by the command ASCII OFF.
Q	Input	Output	Output file will be ASCII. This mode can also be set by the command ASCII ON.
R	Input	Ignored	Input is taken from *name-1* of the @ED control statement; no output text is produced (read-only mode). Use the UP command if you wish to apply changes to your data base.
U	Input (Output if no *name-2*)	Output	Update the symbolic element by applying corrections and create a new symbolic element cycle. For SDF files, the original images will be replaced with the updated ones. The U option functions exactly as it does for SIR$ usage. In particular, an output element name may be specified which need not be the same as the input element; the output element will have the same cycle information as would the updated input element if there were no second element specified. In the case that the current cycle is 62 and the number of cycles in existence is less than the maximum permitted for the element being updated, the output cycle number will be the smaller of *n* and *m-1*, where *n* is the number of cycles in existence and *m* is the maximum number of cycles permitted.
W	Input	Output	Bypass error exit after validity constant check.
X	Input	Output	Take an ERR$ exit upon a fatal or nonfatal error (batch mode only).
Y	Input	Output	If the output file is assigned to another run, you are restricted to read-only mode.
Z			Reserved

2. Edit Mode Commands

2.1. General

Use the Text Editor commands listed alphabetically in 2.2 while in edit mode to modify lines of text in an element or file. If you are in input mode, you can prefix these commands with:

```
@EDIT
```

Some of these commands may be abbreviated to one or two characters as specified. All may be abbreviated to three characters. When in edit mode, type these commands starting on the first character position after the start of entry character (▶).

When using CHANGE, FIND, LOCATE, INSERT, RETYPE, and the corresponding abbreviated commands, leave only one blank space between the command and the parameter image. In all other commands of more than one parameter, leave at least one blank between each parameter. When the Text Editor detects errors in a command, the command does not execute; instead, an error message appears and the Text Editor executes the next command in the runstream (except in the case of the X option in batch mode).

In all command descriptions, the word file is used to mean either file (SDF format) or element interchangeably. Where only a file or element name may appear, the restriction is stated.

NOTE: The commands are shown in uppercase mode; however, the Text Editor will accept ASCII upper- or lowercase or mixed mode for all commands.

2.2. Commands

2.2.1. ADD

Purpose

To add all or portions of a file to the current file.

Format

ADD *name*
ADD *name num1 num2*
ADD+ *name*
ADD+ *name num1 num2*

Parameters

name	specifies the name of the file to be added.
num1 num2	specifies the lines of the file to be added.

Description

The first form of the ADD command adds the entire file, and the second form adds lines *num1* through *num2* to the current file. The lines to be added are inserted at the end of the file unless a "+" immediately follows the command, in which case the lines are inserted following the current position within the edit file. The *name* is the element or file name. If you omit *num2*, the Text Editor adds only a single line at *num1*. The ADD command cannot reference a line number which is greater than 262,143.

2.2.2. APPEND

Purpose

To go to the end of the element or file and enter input mode.

Format

```
APPEND
```

Description

The APPEND command goes to the end of the file and enters input mode, thereby allowing new images to be inserted. You can abbreviate this command to A. Switching to edit mode immediately after APPEND has the same effect as LAST (see conditions listed under LAST).

2.2.3. ASCII

Purpose

To specify that the character set of the output file or element should be ASCII or Fieldata.

Format

ASCII *keyword*

Parameters

keyword	specifies the character set of the output file or element. OFF is the *keyword* for the Fieldata character set. ON is the *keyword* for the ASCII character set.

Description

If this command is not used, the character set of the output is determined from the P and Q options or the character set of the input. In order to have lowercase ASCII representation, you need to use CASE NORMAL in addition to ASCII ON. (See 2.2.5.)

2.2.4. AUTO

Purpose

To specify that you want the file that you are editing automatically saved.

Format

```
AUTO num1
AUTO
AUTO*
```

Parameter

num1	specifies that you want the file that you are editing to be automatically saved for every *num1* number of input or edit transactions.

Description

Use this command to protect yourself against processor or system failure. When an auto save occurs, the Text Editor will display AUTO. If you enter the AUTO command with no parameter, the Text Editor will perform an immediate auto save and reset the input transaction counter. It will also set the auto mode on. The effect of entering the AUTO* command will be the same as entering AUTO, except the frequency will be set to 0 (that is, AUTO* is the same as AUTO followed by AUTO 0).

The Text Editor always saves the file automatically when it goes to the top of the file, and, in this case, "AUTO" will appear if the frequency is nonzero. To cancel auto mode, enter AUTO 0. Note that even if you never used the AUTO command, an auto recovery may be possible because of the implicit auto save each time the Text Editor goes to the top of the file. (Some commands such as MOVE, may pass the top of the file more than once; thus, "AUTO" may appear more than once.)

To recover the contents of the auto save file after a run has terminated abnormally (i.e., by system crash, terminal timeout, loss of carrier, etc.), call the Text Editor with the A option set (see 4.14). If you omit the A option and an auto file exists, you will be asked, DO YOU WANT AUTO RECOVERY? An answer beginning with the letter Y will be assumed to be YES, and the effect will be the same as if the A option had been used. If you enter a Text Editor command, data, or the word NO, the auto file information will be overwritten and lost. The Text Editor will print the name of the file and element (or just file) being edited before asking the question, DO YOU WANT AUTO RECOVERY? to help you decide how to answer the question. If you select auto recovery, the tables of the Text Editor will adjust so that the output file and element will have the name of the file and element in the auto file. The tab character and tab stops will be set to the standard for the element subtype (if any) or to the default standards.

Under some circumstances, the file which was in use at the time of the auto save cannot be retrieved or cannot be restored to the same status. This is the case if the file has read or write keys and the file is not assigned at the time auto recovery is attempted. It will be impossible to exit normally from the Text Editor, so you should free and reassign the file with the proper keys, and again attempt an auto recovery. (Keys are not saved for reasons of security.) In either of these cases, you will be informed by a diagnostic message.

In some of these cases, such as when the R option was used, recovery is impossible; the usual indication of this is an I/O error status 5 and an empty file.

Use the A option to initiate an auto recovery. This option may be useful when you overflow a temporary file (which cannot be expanded) or when you enter OMIT instead of EXIT.

NOTE: *Use the AUTO command sparingly, as it involves extra I/O and computation. Some sites may choose to set a minimum for the line count (larger than the standard of five) for this reason. An AUTO 0 terminates the auto save mode. Entering the AUTO command with no operand causes an immediate auto save to be performed without affecting the auto counter.*

Because the auto save mechanism is designed to recover from abnormal run terminations such as communications failure, it cannot distinguish between such situations and the case of a run terminated by @@TERM, which produces similar behavior in the system. If you customarily terminate your run with @@TERM, you will always find an auto save available when you initiate the next run. This may be wasteful of resources, and it can be avoided by terminating runs with @FIN before doing the @@TERM.

2.2.5. CASE

Purpose

To allow all input lines to be translated to uppercase or left in normal case.

Format

```
CASE UPPER
CASE U
CASE NORMAL
CASE N
```

Description

CASE UPPER causes all input lines to be translated to uppercase. In CASE NORMAL mode, no translation takes place. CASE UPPER is assumed for Fieldata files or elements and when the I option is specified without the Q option (see 2.2.47).

2.2.6. CCHAR

Purpose

To set the continuation character.

Format

CCHAR *char*

Parameter

char	is the character you designate as a continuation character.

Description

When an input line to the Text Editor has this character in it, the Text Editor assumes that the next line or input is a continuation of this current line. This next line will be solicited in the normal manner except that a + will precede the solicitation. The *char* is initially set to a character which cannot be typed in. The *char* can be reset to this nonenterable character by using this command with no *char*. A line may be continued only once.

2.2.7. CHANGE

Purpose

To make corrections in the text by replacing *string-1* with *string-2*.

Format

CHANGE */string-1/string-2/m* G
C */string-1/string-2/m* G
CHANGE */string-1/string-2/>* *n*
C
C?

Parameters

string-1	specifies the desired string in text to be changed. Only the first occurrence of the string will be changed unless the G parameter is specified.
string-2	specifies the string to substitute for *string-1* in text.
>	erases all characters following *string-2* in the text line.
n	specifies the number of lines to be scanned.
G	indicates the global command which changes multiple occurrences of *string-1*.
?	displays the content of the last CHANGE command.

Description

The CHANGE command searches a specified number of text lines for *string-1* (as with LOCATE command, except that CHANGE starts with the current line and LOCATE starts with the following line). When and if the desired string, *string-1*, is found, *string-2* is substituted for it. You can use the transparent change character, TCCHAR (see 2.2.54), in both strings. In *string-2*, it stands for each corresponding occurrence in *string-1*.

The delimiter (/) in the CHANGE command may be any character which does not occur in *string-1* or *string-2* except a blank. The *n* may be omitted, in which case 1 is assumed. Instead of using *n* and G, you may change all subsequent occurrences in the file by using the word ALL (which may be abbreviated A), where *n* is usually specified. Instead of using a large value for *n*, you may specify the word REP (which may be abbreviated R) to indicate that the first occurrence on each line of the rest of the file is to be changed. You can use either *n*, REP, G, or ALL with the > character. If all specifications are omitted, the last CHANGE command will be executed again. Column limits may also be specified.

2.2.8. CHANGE With Column Limits

For the CHANGE (or C) command, column limits must (if used) be the first specification after the string alteration pair, preceding the number of lines to be changed and any specification such as G, A, R, ALL, or REP. For example, the following are acceptable:

```
C /.../.../ [5,10]
C /.../.../ [38] 5
C /.../.../ [39,] G
C /.../.../ [1,50] ALL
```

Example:

All dates marked with an asterisk indicate actual dates.

Use:

```
C /indicate actual dates/.../> [20,]
```

to produce:

All dates marked with an asterisk ...

Immediate column limits, if given, are retained for later use with a CHANGE command with no specifications. (See 4.5 for immediate column limits syntax.)

2.2.9. CPT

Purpose

To print out the Standard Units of Processing (SUPs) used in the present run.

Format

```
CPT
```

Description

The form of the message is (hours)H (minutes)M (seconds)S. The seconds field is given to four decimal places.

Example:

```
CPT
8.8570S
```

2.2.10. CPUNCH

Purpose

To punch parts or all of a file at an onsite or remote card punch.

Format

CPUNCH *num1 num2 device*
CPUNCH *num1 device*
CPUNCH *device*
CPUNCH

Parameters

num1 num2	lines of the file to be punched.
device	specifies the symbiont name to which the output is directed.

Description

After you enter this command, a message (MSG?) will appear. The line you enter will be sent to the system console before the cards are punched. The syntax is the same as the SITE command.

The C option is assumed for the @SYM command by CPUNCH so that you may designate either an onsite punch or a remote punch for the output. See the EXEC Programmer Reference, UP-4144 (see Preface). If you do not specify a device, CP is the default.

If the @SYM operation requested by the SITE or CPUNCH command fails due to an invalid symbiont name, you will be given an additional chance to enter a valid name. For a second invalid name or for any other error, the Text Editor will print the name of the file for which the error occurred, thereby making it possible for you to select an appropriate disposition of the file. At the time the error message is issued, the file created has been freed and cataloged by the Text Editor. Therefore, you should either queue it (via @SYM) for printing/punching using a valid site-id or symbiont name or else delete it from the directory. (See 2.2.41 for printing commands with column limits.)

2.2.11. CSF Executive Control Statement

Purpose

To submit a control statement to the Executive System via the Computer Services Facility (CSF$) Executive Request.

Format

CSF *Executive control statement*

Description

Only statements valid for CSF$ may be submitted. The *control statement* must start (with an at sign [@]) in column 5. The resulting status code is printed.

2.2.12. DELETE

Purpose

To delete lines from the text.

Format

```
DELETE num1 num2
DELETE num1
DELETE+
```

Parameters

num1	specifies the number of lines or the first line of a sequence to be deleted.
num2	specifies the last line of a sequence to be deleted.
+	causes the Text Editor to position itself after the last deleted line.

Description

The first form of the DELETE command deletes lines *num1* through *num2*. The second form deletes the next *num1* lines, starting with the current one. A "+" following the command name will cause the Text Editor to be positioned *after* the lines deleted. This saves one entire pass over the file. You can abbreviate this command to D. The UNDO command will cancel a deletion only if the "+" form was used. The DELETE command cannot reference a line number which is greater than 262,143.

2.2.13. DITTO

Purpose

To duplicate lines in the file.

Format

```
DITTO num1
DITTO num1 num2
```

Parameters

num1	specifies the first line (only line) or the first line of a sequence to be duplicated.

num2	specifies the last line of a sequence to be duplicated.

Description

The DITTO command causes the Text Editor to insert the duplicated lines at the present position in the file. The first form duplicates and inserts the one line at *num1* at the present position. The second form duplicates and inserts all lines *num1* through *num2* at the present position. Be sure the most current line numbers are used. At the completion of the DITTO, the cursor points to the last line inserted; this saves one entire pass over the file. The DITTO command cannot reference a line number which is greater than 262,143.

2.2.14. DOC

Purpose

To add comments or documentation to the existing images.

Format

DOC *lines column* P

Parameters

lines	the number of lines you plan to use for comments starting at the current line.
column	the column at which you wish to insert the comment.
P	if specified, indicates that ". Δ" (period-space) is to appear before the comment.

Description

When you enter the DOC command, each image will be printed to the proper column via ATREAD$. You may then enter your comment or one of the following:

@EOF	– no comment for this line.
@EOF *n*	– the line will be retyped to the specified column plus *n* , where *n* is "1" through "9". This is used for lines of code which extend beyond the normal comment column.
@EOF *x*	– discontinue documentation mode.

After completion of this command, the cursor points to the last line read by the command. The DOC command cannot reference a line number which is greater than 262,143.

2.2.15. EXCH

Purpose

To enter characters not represented in the keyboard character set.

Format

EXCH *char octal-number*

Parameter

char	is the character that is to stand for the number whose internal ASCII representation is *octal-number*

Description

When *char* occurs any place in an input line, it will be replaced by this character. An EXCH with no parameters disables this feature. As an alternative to the *octal-number* for ASCII control characters, the character name (e.g., NUL, BEL, HT, etc.) may be used.

2.2.16. EXIT

Purpose

To exit from the Text Editor.

Format

```
EXIT
```

Description

When you use this command to mark the end of your session with the Text Editor, the Text Editor will apply all corrections to the designated file. When an overflow occurs on EXIT, the Text Editor will automatically expand a cataloged file until it is large enough to hold the output; for temporary files, a message appears, and you may recover the work you have performed by using the A option on the processor call statement.

2.2.17. FC

Purpose

To find characters that correspond exactly column for column starting at column 1 with the *mask* ; the Text Editor flags all occurrences in the remainder of the file.

Format

FC *mask*
FC**n mask*
FC,*n mask*

Parameters

**n*	stops the search after *n* number of occurrences.
,n	limit the search to *n* lines.
mask	is a string of characters that the Text Editor searches for.

Description

If *mask* is omitted, the mask from the last F or FC command is used. You can also use transparent characters in the mask (see 2.2.58). See 2.2.18 for more details.

2.2.18. FIND

Purpose

To find an image that corresponds exactly column for column starting at column 1 with the *mask*. (Only one space must follow this command – leading spaces are significant in *mask*.)

Format

```
FIND mask
F mask
F,n mask
F. mask
F?
```

Description

You can use transparent characters that match any character in the mask. The normal transparent character is a blank, but an alternate may be designated with the TCHAR command. The Text Editor begins its search with the line following the line you are positioned at and proceeds matches a character or reaches the end of the file.

You can abbreviate this command to F. By immediately appending a comma followed by a number onto the command word, you can limit the search to that number of lines. To search only the current line, the command name may be followed immediately by a period. Note that without this option, FIND begins with the following line, not the current line, and repeats. If *mask* is omitted, the mask from the last F or FC command is used. F? will print the saved mask.

Both the FIND and the FC commands recognize the TAB character. When the Text Editor encounters a TAB character, all characters are assumed to match up to the next tab stop, and matching resumes at that point. This can save typing and counting when working with column-formatted information.

2.2.19. GI

Purpose

To position the cursor at a line numbered according to the input text, even after the input file has been edited with insertions, deletions, and other modifications.

Format

```
GI number
```

Parameter

number	is the line number of the text before the input was modified.

Description

If the desired line has been deleted or altered in some way, the message:

```
REQUESTED LINE DELETED OR ALTERED
```

appears and the cursor points to the next existing line. This command is useful when you are editing text while working from a listing. You cannot use this command with print files.

2.2.20. IB

Purpose

To insert a line of text before the current line.

Format

IB *text*

Parameter

text	is the *text* that you wish to insert.

Description:

This command behaves exactly the same as the INSERT command, except that the line is inserted before the line you are positioned at instead of after it. If *text* is blank, the Text Editor inserts a blank line.

2.2.21. INLINE

Purpose

To allow inline editing of a given line.

Format

INLINE *number term-sub*
IN *number term-sub*
IN *number*
IN *term-sub*
IN+
IN

Parameters

number	specifies the line to be edited.
term-sub	is the termination character for editing commands.

Description

If you do not specify the *number*, the current line is the one to be edited, unless the command is followed by a "+", in which case the next line will be the one to be edited. Otherwise the Text Editor proceeds to line *number* and displays this line. You can then enter the INLINE editing information directly below the line to modify it. The INLINE command cannot reference a line number which is greater than 262,143.

The *term-sub* field is optional; the normal termination character for editing commands is the exclamation point ("!"), but you can use this field to specify a different character if the information to be edited contains exclamation points. (Note that if you omit the *number* field, the *term-sub* field must not be a character that is interpretable as an expression.) The INLINE editing characters to be used are:

I – The string following this command is inserted following the character immediately above the I. The string is delimited on the right by the termination character "!".

R – The characters following the R will replace the characters immediately above them. A "!" is required to terminate replacement (unless *term-sub* is used).

D – The characters in the line above are deleted between D and the "!".

You can use more than one of the insert, delete, and replace operations in a single INLINE edit. The letters I, R, and D may be entered in either upper or lowercase. Before using this command in @@CQUE mode on a TTY terminal, enter the command OFF U; otherwise, the alignment of the editing line will be incorrect.

The alternate character specified by *term-sub* remains in effect for only a single command.

Example 1:

```
INLINE 27 *
+++\c1 <d EXEC Summary> 542A64 \c2 <d 7824> 71 \c3 39R1
+▶     I[text inserted]*
```

produces:

```
\c1 <[text inserted]d EXEC Summary> 542A64 \c2 <d 7824> 71 \c3 39R1
```

Example 2:

```
INLINE 27 *
+++\c1 [text inserted]<d EXEC Summary> 542A64 \c2 <d 7824> 71\c3 39R1
+▶        rxxxx     xx*
```

produces:

```
\c1 [texxxxx     xx<d EXEC Summary> 542A64 \c2 <d 7824> 71 \c3 39R1
```

Example 3:

```
INLINE 27 *
+++\c1 [txxxx     xx]<d EXEC Summary> 542A64 \c2 <d 7824> 71 \c3 39R1
+▶      d           *
```

produces:

```
    \c1 <d EXEC Summary> 542A64 \c2 <d 7824> 71 \c3 39R1
```

2.2.22. INPUT

Purpose

To direct the Text Editor to enter input mode.

Format

```
    INPUT
```

Description

In this mode everything that you type in is inserted in the file until you exit from the mode. This is especially useful when you are entering large volumes of input. To exit from this mode, type an @EOF when in EOF mode (see ON and OFF commands), a carriage return (blank line) when not in EOF mode, or an @EDIT when in either mode. The INPUT mode recognizes tabs and the MSCHAR.

2.2.23. INSERT

Purpose

To insert a line of text following the current line.

Format

INSERT *text*
I *text*
I+

Description

When you enter this command and follow it with the text you wish to insert, the new line will then be the point at which the cursor is positioned. Insert your text after the first blank following the INSERT command, which may be abbreviated to "I." If a "+" immediately follows the command, you may enter the text on the next line (this provides more room).

2.2.24. LAST

Purpose

To move to the last line in the file while in edit mode.

Format

```
    LAST
```

Description

The last line cannot be printed or altered after this command until the file position (line number) indicated as LAST is reentered. There are six commands which are illegal after the LAST command has been used (until a line has been added or the file position is changed); these are CHANGE, DELETE, DOC, IB, INLINE, and RETYPE. An attempt to use any of these immediately after entering the LAST command will produce a diagnostic message (and error termination in batch mode if the X option was set).

In most cases, any command that attempts to move to the current line number will simply cause the current line to be typed. Because of the special situation which exists after the LAST command has been used, the Text Editor will not allow you to modify the last line until you request that line by number. By entering the last line number, you can remove the restrictions created when you used the LAST command.

2.2.25. LC

Purpose

This command behaves as the LOCATE command (see 2.2.28), except that it locates all occurrences of the string in the remaining text.

Format

LC *string*
LC *quote-char string quote-char*
LC∗*n string*
LC,*n string*

Parameters

string	is the set of characters that the Text Editor will locate.
quote-char	is the character defined by LCHAR. The default is a single quote (').
∗n	stops the search after *n* number of occurrences.
,n	is the search is limited to *n* lines.

Description

The Text Editor will print the line number just before each line containing an occurrence. If you append an asterisk to the LC command, followed by a number, the Text Editor will stop searching after that number of occurrences of the string are found. A comma followed by a number indicates the number of lines to search. If you omit *string*, the string from the last L or LC command is used. Column limits may be specified (see 4.5).

2.2.26. LCHAR

Purpose

To set the quote character for the LOCATE command.

Format

    LCHAR *char*

Description

The default character is a single quote ('). LCHAR without a *char* will result in no character being represented as LCHAR.

2.2.27. LIMIT

Purpose

To set the left and right column limits for the CHANGE, LOCATE, and PRINT commands.

Format

    LIMIT *keyword num1 num2*

Parameters

keyword	can be CHANGE, LOCATE, or PRINT (each of which may be abbreviated to its first letter).
num1 num2	are column numbers that specify the left and right column limits.

Description

If you specify only one column number, it designates the right limit, with 1 assumed as the left limit. If you do not specify any column numbers, the column limits are set to the default values (1,132). If the keyword is CHANGE, then this command sets limits on the columns which will be searched by the CHANGE command. This is useful for protecting areas of text lines in a file. If the keyword is LOCATE, then this command sets limits on the columns which will be searched by the LOCATE and LC commands. If the keyword is PRINT, then this command sets limits on the columns which will be printed by the output commands (PRINT, LNPRINT, QUICK, LNQUICK, PUNCH, CPUNCH, SITE, and LNSITE), as well as by other commands which print lines of text. The print column limits specified by the user are rounded to the nearest ASCII word boundary, e.g., LIMIT PRINT 8 9 will cause columns 5 to 12 (words 2 and 3) to be printed by the PRINT command (see 2.2.41).

Limits specified by this command may be overridden on any single command by the use of immediate column-limit specifications (see 4.5).

2.2.28. LOCATE

Purpose

To search the text for a given string of characters.

Format

LOCATE *string*
LOCATE *quote-char string quote-char*
LOCATE,*n string*
LOCATE. *string*
L *string*
L ?

Parameters

string	the set of characters that the Text Editor will locate.
quote-char	the character defined by LCHAR.
,n	indicates that the Text Editor is to only search *n* number of lines to locate the *string*.
a period (.)	indicates that the Text Editor is to only search the current line to locate the *string*.

Description

The Text Editor searches through the text beginning at the line following the line at which the cursor is positioned and proceeds sequentially through the text until it finds the string or until it reaches the end of the file. The first form ignores multiple blanks in the images. The second form requires that the text image be exactly the same as the string within the two quote characters. You can abbreviate this command to L. If you append a comma followed by a number onto the command word, the Text Editor will limit the search to that number of lines. To search only the current line, place a period immediately after the command. When you do not follow the command with a period, the search begins with the following line and not the current line. If you omit the *string*, the string from the last L or LC command is used. If the LC variable is to be referenced after a LOCATE, the quoted form should be used; otherwise, the position indicated may include leading blanks. See 4.5 for use of column limits. Leave only one blank between LOCATE and *string*. The form "L?" will print the last LOCATE string.

2.2.29. LOCATE With Column Limits

For the LOCATE (or L) and LC commands, if immediate column limits are specified, they must immediately follow the command and any count specification, with no intervening blanks. At least one blank must separate the column limits from the LOCATE target string. For example, any of the following are acceptable:

```
L[3,5] ...
L.[3,5] ...
L,100[8,18] ...
LC*3[10,21] ...
```

Immediate column limits are not saved for later use on a LOCATE command with no target; they must be entered each time. (See 4.5 for immediate column limits syntax.)

2.2.30. MAIL

Purpose

To send messages to, and to receive messages from, other users.

Format

```
MAIL user-id
MAIL?
```

Parameter

user-id	the person's *user-id* that you are sending a message to.

Description

The Text Editor will solicit up to 100 lines of input with:

```
MAIL**
```

If the message you are sending is less than 100 lines, enter @EOF to conclude your message. After the designated person receives the message, the message is deleted.

The Text Editor does not search for mail in batch mode; therefore, there will never be a solicitation:

```
DO YOU WANT YOUR MAIL?.
```

In demand mode, the Text Editor will search for mail on entry (after the sign-on line) in edit mode. In input mode, the search will be made when you switch to edit mode. The form MAIL? will search for mail without further query.

If your response to the solicitation:

```
DO YOU WANT YOUR MAIL?
```

comes from an add file or does not begin with "Y" and is not "NO", then it will be treated as a command.

NOTE: The Text Editor will look for mail only the first time it is called in a run. The system generation parameter MAILINIT controls whether the Text Editor will look for mail each time it is called. As released, the value is 0. Setting it to 1 will set this mode of operation. A system generation parameter MAXMAIL is provided to control the size of mail files. To encourage larger (and thus fewer) files, the Text Editor is released with this parameter set to 100.

2.2.31. MAXLINE

Purpose

To set the maximum length (1 - 132) to which a line may increase.

Format

MAXLINE *number*

Parameter

number	is the maximum length to which a line may increase.

If the maximum length is exceeded, the line will be truncated. The default is 132. The largest acceptable line length is configurable in the Text Editor but also depends on what is permitted by the operating system.

2.2.32. MCCHAR

Purpose

To specify a character that separates multiple commands given on a single line.

Format

MCCHAR *char*

Parameter

char	is the character that you designate as the command separator.

Description

Once you've specified the character *char* to separate multiple commands, each occurrence of the multiple command separator character terminates a command or input line and begins a new one. For example, you may enter:

```
MCCHAR #
GI 453 #C /ABC/XYZ/
```

to make changes on line 453. If the last nonblank character on a line is the MCCHAR character, a blank line will follow the last command on a line.

Or you may enter:

```
MCC %
L 35R1 %c /35R1/36R1/
```

This will locate 35R1 and then change 35R1 to 36R1.

When using the LPTST command (see 3.4) with this feature, the skip count is decremented by 1 for each line (NOT command) scanned, including the remainder of the line containing the LPSUB command, if there were additional commands on that line. The LPSUB command substitution count applies to commands rather than lines. Care is required, however, if the substituent contains the MCCHAR character. In order to be recognized, labels (*xxxx*) must appear as the first command on a line.

This feature is normally disabled, and it may be discontinued by entering MCCHAR with no character present. This character is also recognized in input mode and allows entry of several text lines in one line of input. In input mode, each command entered using the @EDIT feature must begin with @EDIT if more than one command appears on a single line through use of the MCCHAR character. Since the @EOF image is a system image rather than an ED image, it must always appear by itself on a single line. MCCHAR may not be used in a LOOP or MACRO.

2.2.33. MOVE

Purpose

To move lines from the original position to the line following the line that the cursor is positioned at.

Format

MOVE *num1*
MOVE *num1 num2*

Description

This command performs the same operation as the DITTO command except the original lines are deleted after the duplication has taken place. The syntax is the same as for the DITTO command. Be sure the most current line numbers are used. At the completion of the MOVE, the cursor points to the original line number. The MOVE command cannot reference a line number which is greater than 262,143.

2.2.34. MSCHAR

Purpose

To set a character which will be translated to an at sign (@) when it is used in column one of input lines in input mode.

Format

MSCHAR *char*

Parameter

char	is the character that you will use to represent an at sign (@) in column 1 of an input line.

Description

If *char* is blank, this command will not work. You can use MSCHAR to append comments to FORTRAN statements when using the DOC command.

2.2.35. Number

Purpose

To position the cursor at a desired line in the text.

Format

> *number*
> +*number*
> -*number*

Description

number is the line number that you wish to position the cursor to. +*number* is the number of lines you wish to move ahead to. When you use the format -*number*, you are specifying the number of lines that you wish to move back to. When the specified line is located, the Text Editor displays it (if not in BRIEF mode), and modifications may be made to it. If you desire to insert lines before line 1, type in 0. This will position the cursor immediately before the first line. A number cannot reference a line number greater than 262,143.

2.2.36. OMIT

Purpose

To be used if you do not want your corrections applied to the file.

Format

> OMIT

Description

The file will remain as it was at the beginning of the editing session. In read-only mode, EXIT is synonymous with OMIT.

2.2.37. ON and OFF

Purpose

To define special modes within the Text Editor. ON turns the mode on, and OFF turns it off.

Format

> ON *special mode,...,special mode*
> OFF *special mode,...,special mode*

Parameters

The *special modes* are:

BRIEF	Do not print images from file unless specifically directed by PRINT, QUICK, etc. This mode is also controlled by the N option.
DSPLIT	Delete lines transferred by SPLIT command.
EOF	A special mode where blank lines may be entered. An INP command initiates input mode and an @EOF exits from input mode to edit mode. While in input mode, blank lines may be entered. Also the INSERT command with no image following will enter a blank line. To terminate the EOF mode, use OFF EOF.
INPSEQ	When on, input solicitation (if active) will include the input line number in parentheses. (This is the line number referenced by the GI command.)
LSTINP	When on, input lines and commands to the ED processor will be printed in the run listing. When off, input will not be listed. This mode is also controlled by the L option on the ED processor call statement. This mode may be turned on to trace the statements executed by a LOOP or MACRO call.
MEMORY	Remember modes on successive executions within a single run.
NUMBER	Precede each line printed out with its number.
PCNTRL	Print control images will be printed if this mode is on.
QUICK	Compress extra blanks from all output, except for PRINT command.
TRDINP	When on, demand input is via TREAD$. When off, demand input is via READ$. There is no effect in batch mode. This may be turned off in @@CQUE mode to permit the fastest possible typing speed.
UNISCP	Allow correct character placement on UNISCOPE terminal with the INLINE command, or on other devices in @@CQUE mode.
XBRIEF	Do not echo lines transferred by SPLIT or ADD.

All of the modes may be abbreviated to the first character.

Description

You may choose any of these special modes while your session is in progress. Use the ON command to select the special modes you desire. Use the OFF command to turn off one or more special modes.

2.2.38. OPR

Purpose

To send a message to the system console operator.

Format

```
OPR string
OPR* string
```

Parameter

string	is the message (which may not be more than 50 characters).

Description

The first form sends the message *string*. The second form does the same, but also solicits an answer. The string may not be more than 50 characters or it will be truncated. You can read the operator's response via the LPSUB and LPTST OP specification.

2.2.39. PCC

Purpose

The Print Control Character (PCC) command behaves like the PRINT command, including the use of the form PCC+.

Format

```
PCC num1 num2
PCC num1
PCC+
PCC!
```

Parameters

num1 num2	prints lines *num1* through *num2*.
num1	prints the next *num1* lines.
+	starts the print with the next line.
!	prints the entire file from the top.

Description

Any ASCII control characters on a line will be mapped into the character whose code is 0100 larger, and any lowercase characters on a line will be mapped into the character whose code is 040 smaller. The line will then be printed, followed by a second line which will contain spaces below any characters which were in the 64-character ASCII set, L below any characters which were lowercase, and C below any characters which were control characters. Thus, if a line consists of the characters uppercase a, lowercase n, and BEL, PCC would print that line as:

```
ANG
 LC
```

2.2.40. PRINT

Purpose

To print out lines of text.

Format

```
PRINT num1 num2
PRINT num1
PRINT!
PRINT+
```

Parameters

num1 num2	prints lines *num1* through *num2* (maximum of 262,143 lines).
num1	prints the next *num1* lines.
+	starts the print with the next line instead of the current line.
!	prints the entire file from the top (maximum of 262,143 lines).

Description

If the command is immediately followed with a "+" the printing starts with the next line instead of the current one (example: PRINT+3). The command followed by an exclamation point prints the entire file from the top. If no number or recognizable symbol follows the command, a 1 is assumed; that is, the present line will be printed. This command may be abbreviated to P. (See 2.2.39 and 2.2.41.)

2.2.41. Output Commands with Column Limits

Column limits may be used with the following printing commands: PRINT, P, PCC, QUICK, Q, OUTPUT, O, SITE, PUNCH, and CPUNCH. The immediate column limits, if specified, must be separated from the command (and its trailing delimiter, if any) by at least one space and must precede any line number specifications. For example, the following are all acceptable (the P command will be used for purposes of illustration, but any of the commands mentioned above may be used):

```
P[5,10]
P+ [11,21]
P! [38]
P [15,35]
P [39,]11,20
```

Note that as for the LIMIT P case, column limits for printing will be adjusted to word boundaries. That is, a left limit is reduced to the next smaller column, which causes it to fall on a word boundary, and a right column is increased to the next larger column, which causes it to fall on a word boundary. For example, the limits pair [12,22] would have the same effect as the pair [9,24]. (See 4.5 for immediate column limits syntax.)

2.2.42. PUNCH

Purpose

To punch paper tape for input at a terminal that has punch and read hardware.

Format

PUNCH *num1 num2*
PUNCH *num1*
PUNCH!
PUNCH+

Parameters

num1 num2	punches lines *num1* through *num 2*.
num1	punches the next *num1* number of lines.
!	punches the entire file
+	starts punching, starting with the next line instead starting at the current line.

Description

The syntax for this command is the same as that for the PRINT command. When you enter the command, the following response will appear:

```
DEPRESS PUNCH ON
```

The processor will then wait for you to push the punch ON button on the paper tape punch hardware. After pausing, the designated lines will be typed out, which will cause the paper tape to be punched at the same time. Rubouts will be punched at the start and end of the tape. You can use a tape so produced as normal paper tape input. (See also 2.2.41.)

2.2.43. QUICK

Purpose

To print lines with all nonsignificant blanks omitted.

Format

```
QUICK num1 num2
QUICK num1
QUICK!
QUICK+
```

Parameters

num1 num2	prints lines *num1* through *num2*. (Maximum of 262,143 lines.)
num1	prints the next *num1* lines.
+	starts the print with the next line instead of the current line.
!	prints the entire file from the top. (Maximum of 262,143 lines.)

Description

This command provides a fast method of examining areas of the file. Parameters *num1* and *num2* are the same as on the PRINT command. Plus (+) may also be used on the second form with the same meaning. You can abbreviate this command to Q. (See also 2.2.41.)

2.2.44. REMARK

Purpose

To include commentary in loops and to allow LOOPs and MACROs to request input, although it may be useful in an @ADD file.

Format

```
REMARK text
REMARK*text
REMARK!text
REMARK?text
```

*NOTE: Text begins with the first character after *, !, ?, or space and ends with the last nonspace character.*

Description

The *text* may consist of any comment you wish to include. If you use the form REMARK*, the text will be printed prefixed REM*:, which may be useful in conjunction with the XTI command. The form REMARK! has the same effect as REMARK*, except that the prefix will not be printed. The form REMARK? is used to solicit a response; this response may be accessed later via the LPSUB and LPTST QU specification. The LPSUB command can make substitutions into the *text* field of a REMARK command.

2.2.45. RETYPE

Purpose

To replace text in the current line with the string of text following the first blank after the command.

Format

```
RETYPE string
R string
```

Description

A "+" may be used after the command with the same meaning as with the INSERT command.

2.2.46. RP

Purpose

To set a repeat counter for the INSERT command.

Format

```
RP number
```

Parameter

number	is the number of times you want the insertion repeated.

Description

Any insertion will be repeated *number* times. The counter remains in effect until explicitly reset to 1. If you omit *number*, 1 is assumed as the default.

2.2.47. SCALE

Purpose

To print a line that can be used as a measuring scale for column sensitive operations.

Format

```
SCALE
SCALE n1
SCALE n1 n2
SCALE n1 n2 n3
SCALE! n1 n2 n3
SCALE!
```

Description

The scale consists of the digits 1 through 9 repeated with the zero (0) digit displayed every tenth digit. Columns *n1* through *n2* are displayed. If *n2* is omitted, 72 is assumed; if *n1* is omitted, 1 is assumed. If the command is entered in the form SCALE!, a second line will be printed, consisting of the tens digits. If both *n1* and *n2* are present, there may be a third parameter, *n3*, which indicates an offset value to be added to the digit printed in each column. You can use this to line up columns for input with the I command, by entering SCALE 8 80 3, where the actual values used will depend on how many digits are printed by input solicitation. If *n3* is omitted, a value of zero is assumed.

Examples:

```
SCALE
123456789012345678901234567890123456789012345678901234567890123456789012

SCALE 2
 23456789012345678901234567890123456789012345678901234567890123456789012

SCALE 10 40
         0123456789012345678901234567890

SCALE 10 40 2
         2345678901234567890123456789012
```

2.2.48. SEQ

Purpose

This command numbers, in sequence, a specified set of columns on each image in the file.

Format

SEQ.*id i,j*
SEQ.*id col i,j*

Parameters

id	indicates the id-field.
i	specifies a value indicating the starting number.
j	indicates the increment.
col	specifies the column limits of the sequence field.

The maximum value of *i* or *j* is 262,143.

Description

Omitted values in the SEQ command are given a default of 100; if *i* is omitted, *j* will be ignored.

The id-field may contain any ASCII alphanumeric characters. If it is omitted, the sequence field will contain only the sequence number. Sequence numbers are printed with leading zeros. If the sequence number is or becomes too large to fit in the field, it will be reduced modulo the appropriate power of ten. If the id-field is the same size as the sequence field, no sequence numbers will appear, just the *id*. If the id-field is larger than the sequence field, an error will result.

The *col* parameter defines the column limits of the sequence field in the format for immediate column-limit specifications on the CHANGE command ([*m,n*], etc.). If the *col* parameter is omitted, columns 73 through 80 are assumed. The use of a column specification may be particularly helpful when editing COBOL or BASIC elements, to create required sequencing or line numbering.

2.2.49. SET

Purpose

To set the tabs to be used with the following commands: INSERT, IB, RETYPE, FIND, FC, and input mode.

Format

SET *tab1 tab2 tab3 ... tabn*
SET+ *tab1 tab2 tab3 ... tabn*

Description

You can set as many tabs as you desire. Each SET command redefines all previous tabs, so a SET with no tabs clears the tabs. List tabs in ascending order.

If you have not used the SET command, the tabs are set as follows:

7,73	if the input element type is FORTRAN or FORTRAN proc,
8,12,73	if the input element type is COBOL or COBOL proc,
6,11,16,21,26	if the input element is type ALGOL, PLUS or PL/I,
11,21,39,73	otherwise.

If you enter the command as SET+, the indicated tabs will be added to the set of tabs already in use. In this case, the first tab specified must be larger than the largest existing tab. You can use this feature in a loop to set up a number of evenly spaced tab stops as follows:

```
SET
LOOP 15 6 5
LPSUB L,$
SET+ $
@EOF
```

The first SET command is necessary to clear any existing stops. The loop given will set tab stops at columns 6, 11, 16, 21,...,76.

2.2.50. SHCHAR

Purpose

To specify a character for case shifting for input from devices (teletypewriters, card readers) that do not have lowercase capabilities.

Format

```
SHCHAR char
```

Parameter

char	is the character you designate as the shift character.

Description

When the specified *char* is encountered on any line processed by the Text Editor, all following uppercase alphabetic characters will be translated to the corresponding lowercase characters until another shift character is encountered. You should always specify CASE NORMAL before enabling this feature. If you do not specify *char* , the shift feature does not work. The printout from the STATUS command will indicate whether shifting is active or not. The shift character itself will be deleted from any lines in which it appears.

2.2.51. SITE

Purpose

To direct output to an onsite or remote printer.

Format

```
SITE num1 num2 device
SITE num1 device
SITE device
```

Parameters

num1 num2	prints lines *num1* through *num2* .
num1	prints the next *num1* lines.
device	specifies the symbiont name of the device that will be used for printing.

Description

The meanings for *num1* and *num2* are the same as for PRINT, except that if no numbers are given, the third form is assumed. The *device* form specifies the symbiont name to which output is sent. If the field is not specified, PR is assumed. After this command is entered, the following message will be displayed:

```
HDG?
```

The response will be used to head the onsite output. Periods must not be used in this header as anything beyond the period will not be printed. The EXEC accepts only a valid print control function following a period. After the output, the following message will be displayed:

```
MSG?
```

You should enter the information necessary to indicate where and to whom the output should be returned.

See 2.2.41 for a discussion of printing commands with column limits. The SITE command cannot reference a line number which is greater than 262,143.

2.2.52. SPLIT

Purpose

To build new elements or files from portions of a current file.

Format

SPLIT *name*
SPLIT *name num1 num2*
SPLIT! *name*

Parameters

name	is the name of the file that will contain the reproduced text.
num1	is the number of the line where reproduced lines begin.
num2	is the number of the line where reproduced lines end.
!	indicates that you want the whole file copied.

Description

The first form causes all lines preceding the current line to be reproduced as the designated file. The second form causes lines *num1* through *num2* to be reproduced. An "!" immediately after the SPLIT command causes the whole file to be copied. The character set (ASCII or Fieldata) of the new file is the same as the character set of the original file.

NOTE: The default for the special mode DSPLIT is ON and data lines transferred by SPLIT will be deleted.

When used in read-only mode, the SPLIT command will always function as if DSPLIT mode is off. If DSPLIT is on, a diagnostic message will appear. The SPLIT command cannot reference a line number greater than 262,143.

2.2.53. SSP

Purpose

To override the automatic value of 1 for line spacing for newly inserted lines or to display the current line spacing value for a line.

Format

```
SSP n
SSP?
```

Description

The SSP command has two formats. The form SSP? prints out the line-spacing value for the current line. The form SSP *n* sets the line-spacing value for the current line to *n*; if *n* is omitted, a value of 1 is used. This command may only be used when editing print files. Only the SSP? form is valid in read-only mode.

2.2.54. STATUS

Purpose

To request the status of special modes set by the ON and OFF commands.

Format

STATUS *special mode,...,special mode*
STA*

Parameters

The *special modes* are:

BRIEF	Do not print images from file unless specifically directed by PRINT, QUICK, etc. This mode is also controlled by the N option.
DSPLIT	Delete lines transferred by SPLIT command.
EOF	Special mode where blank lines may be entered. INP command enters input mode and @EOF exits from input mode to edit mode. After the @EOF, OFF EOF is necessary to get out of EOF mode. While in input mode, blank lines may be entered. Also, the INSERT command with no image following will enter a blank line.
INPSEQ	When on, input solicitation (if active) will include the input line number in parentheses. (This is the line number referenced by the GI command.)
LSTINP	When on, input lines and commands to the ED Processor will be printed in the run listing. When off, input will not be listed. This mode is also controlled by the L option on the ED Processor call statement. This mode may be turned on to trace the statements executed by a LOOP or MACRO call.

MEMORY	Remember modes on successive executions within a single run.
NUMBER	Precede each line printed out with its number.
PCNTRL	Print control images will be printed if this mode is on.
QUICK	Compress extra blanks from all output, except for PRINT command.
TRDINP	When on, demand input is via TREAD$. When off, demand input is via READ$. There is no effect in batch mode. This may be turned off in @@CQUE mode to permit the fastest possible typing speed.
UNISCP	Allow correct character placement on UNISCOPE terminal with the INLINE command, or on other devices in @@CQUE mode.
XBRIEF	Do not echo lines transferred by SPLIT or ADD.

All of the modes may be abbreviated to the first character.

Description

If no *special modes* are specified, the status of all will be listed, along with other status information, as shown below in the example.

The form STATUS* with no specifications (see example) will not list the mode values, but will only give the information shown in the example.

Example:

```
0:▶STA*

TAB:; CCHAR:NONE LCHAR:" MSCHAR:NONE TCHAR:SP TCCHAR:NONE SHCHAR:NONE
EXCHAR:NONE MCCHAR:NONE TABS:11,21,39,73 MAXLINE:132 CASE:NORMAL MODE:ASCII
SHIFT:OFF AUTO:0 LIMITS - CHANGE:1,132 LOCATE:1,132 PRINT:1,132 LOOP SPACE: 488
WORDS
```

2.2.55. STK

Purpose

To provide a way of remembering the set of ON/OFF modes.

Format

STK *x*

Description

The stack depth limit is five. If *x* is omitted or is UP, the modes will be saved. If *x* is DN or DOWN, the modes will be restored. Other specifications are erroneous. No checks are made for the number of UP operations exceeding five or the number of DN operations exceeding the number of UP operations. It is intended primarily for use in macros.

2.2.56. TAB

Purpose

To specify which character is to be used as a tabulator character.

Format

```
TAB tab-char
TAB
```

Parameter

tab-char	is the character you choose to function as the tab character.

Description

This character is recognized on the INSERT, IB, RETYPE, FIND, and FC strings and is recognized on all input when in the input mode. The character is not transmitted to the file and behaves just as a tab on a typewriter. If you do not use the command to define a tab character, the *tab-char* is set as follows:

"#"	(number sign) if the input element is type ALGOL, BASIC, or PLUS.
" " "	(quotation mark) if the input element is type PL/I.
"_"	(underline) if the input element is type DOC.
";"	(semicolon) for all other element types, and files.

You can also use a blank as a tab character by specifying the *tab-char* as SP or BL. ASCII control characters may be specified by name (NUL, SOH, etc.) as well as by direct entry. If you enter this command without specifying a tab character, the command will not work. For example: TAB = ? allows you to use the question mark as a tab character.

2.2.57. TCCHAR

Purpose

To set a transparent character for the CHANGE and LOCATE or LC commands.

Format

```
TCCHAR char
```

Description

If specified in *string-1* of the CHANGE command or *string* of LOCATE/LC, the character *char* will match any character in the text. There may be any number of occurrences of *char* in *string-1* of a CHANGE command. If the transparent character occurs in *string-2* of the CHANGE command, each occurrence of the character will stand for the character in the original image matched by the corresponding occurrence of the transparent character in *string-1* . If the number of transparent characters in *string-2* exceeds the number in *string-1* , the excess transparent characters in *string-2* will stand for themselves.

For example, the command sequence:

```
TCCHAR %
C /%/% / G
```

will place a space after each character of the current line.

If the transparent change character field is a space, this feature is disabled.

TCCHAR or TCCHAR SP sets the transparent change character to a space.

The feature is initially disabled. The TCCHAR will also be recognized as a "match anything" character for the LOCATE and LC commands.

2.2.58. TCHAR

Purpose

To set a transparent character for the FIND command.

Format

TCHAR *char*

Description

This command sets a transparent character for the FIND command. Omitting *char* disables the feature. TCHAR BL resets the default to blank.

2.2.59. TIME

Purpose

To print out the date, time, and cycle information and the name and type of the output element or file.

Format

    TIME

Example:

```
ELT THU-10/12/78-13:13:38-(8,9)
KMM*TEXT(1).ED(9)
```

2.2.60. TYPE

Purpose

To specify the processor type for symbolic element output.

Format

    TYPE *processor-mnemonic*
    TYPE* *processor-mnemonic*

Parameters

processor-mnemonic	is the mnemonic or the octal number for the processor type.
*	indicates that you want the tab character and the tab settings to change to accommodate the type.

Description

The processor mnemonics of the TYPE command are: ALG, APL, APT, ASM, ASMP, BAS, COB, COBP, DOC, ELT, FOR, FORP, FLT, LSP, MAC, MAP, MSM, MSD, PLS, PL1, PNC, SSG, SEC, TCL, PGA, QLP. You can also use octal numbers instead of the mnemonic.

If you enter the TYPE command as TYPE* *xxx*, then the tab character and tab settings appropriate to the type *xxx* will be established, replacing any tab character and settings currently in effect.

2.2.61. UNDO

Purpose

To make the Text Editor ignore all the changes made since the last return to the top of the file and to position the cursor at the top of the file.

Format

    UNDO

Description

If you haven't made any changes, a diagnostic message will appear, and the position of the cursor remains unchanged. Since some commands (for example, DELETE, MOVE, and DITTO) implicitly go to the top of the file, it will not be possible to undo their effects.

2.2.62. UP

Purpose

To cause the Text Editor to behave as if the U option had been specified on the control statement; in other words, it allows you to update the file or element.

Format

```
UP
```

Description

Use this command if the entry to the Text Editor was made with an R option (@ED,R).

2.2.63. WAIT

Purpose

To make the Text Editor wait a specified amount of time or until you give it a command to resume processing.

Format

WAIT *n*
WAIT UNTIL *time*

Parameters

n	is the time in seconds (decimal) that you want the Text Editor to wait. The form *n* H *n* M *n* S indicates the time in hours, minutes, and seconds.
time	is the time of day that you want to resume editing with the Text Editor. You can use the keywords AM or PM or you can use the 24-hour clock notation.

Description

If you use @@X C to interrupt the waiting, the interrupt will occur after at most a 30 second delay. The time may be specified as a decimal number of seconds or by parameters of the form *n* H, *n* M, and *n* S, where any combination is permitted, *n* is a decimal integer of six digits or fewer, and values of *n* need not be less than 60. The keyword UNTIL specifies a time of day to be waited for. The keywords AM or PM may be used, or the 24-hour clock.

For example:

```
wait 10s
```

delays processing for 10 seconds and

```
WAIT UNTIL 10:09 AM
```

will delay processing until the time 10:09 is reached.

2.2.64. XPC

Purpose

To submit a print control image via APRTCN$.

Format

XPC *print control image*

Description

The *print control image* must start in column 5 and may contain one or more print control functions, limited only by the length of the line.

3. Loop Operations

3.1. LOOP

Purpose

To allow repetitive execution of a group of statements.

Format

```
LOOP n start increment
LOOP! n start increment
LOOP?
LOOP n
LOOP+
LOOP*
```

Parameters

n	specifies the number of times the statements are executed (default value is 0).
start	specifies the initial value of the counter (default value is 1).
increment	indicates that the counter is to be incremented by this amount on each iteration (default value is 1).

Description

You should enter the LOOP *n* command where *n* is the number of times the commands are executed. The commands to be executed will then be solicited with an LP**.

The LOOP command ends with an @EOF (with nested loops, only the outermost loop is ended with @EOF; LPEND will end inner loops). If you want to place an @EOF in the loop, use the @EOF L command. The sentinel character L will allow the LOOP entry to continue. To stop an erroneous loop before execution begins, enter @@X C followed by @EOF. Normal execution will stop after *n* executions, when the bottom of the file or element is passed, or if you enter @@X C.

Some commands, such as LOCATE and FIND, will stop the execution of the loop when they reach the end of the file. However, if you enter LOOP*n* ! (which is an incorrect format), you will cause a nonstop operation; the execution will not stop at the end of the file (@EOF).

The LOOP? (no parameters) format will print out the currently stored loop. The LOOP+ (no parameters) format will expand the space available for storage of loops and macros by 512 words; you may need this for exceptionally large loops or to store many macros. You can use LOOP+ more than once. Use the LOOP* format to contract the space used for loop and macro storage when it is no longer needed. If you attempt to contract to less than the initial size, your request will be ignored. In order to release space, there must be no information stored in the area to be released, or else the command will be ignored. This means that you must delete (using MAC*) macros that are no longer in use before attempting to release storage. LOOP! causes the currently stored loop to execute again with new parameters.

You may nest loops up to ten deep. Terminate all nested loops using the LPEND command. The LOOP command is identical for nested loops, except that the exclamation point in LOOP! will be ignored, as it is meaningless. Macro calls are treated as nested loops.

Within a loop, it is possible to move backwards through the element (for example, using the commands T, -n, 0). A loop is terminated by a command which reaches the end-of-file. The SPLIT command will not stop a loop. The FC and LC commands, as well as unsuccessful FIND and LOCATE commands that reach the end-of-file, will stop a loop. The DELETE, MOVE, and DITTO commands are treated as special cases and will not terminate a loop. The LAST command goes to the end-of-file, but does not pass to the start of the file, and thus will not stop a loop.

Example:

```
0:▶LOOP 99999
LP**▶LOCATE ABC=
LP**▶I  PRINT ABC
LP**▶@EOF
```

This loop will locate all assignments to the variable ABC and insert a print statement following the assignment statement.

3.2. LPJUMP

Purpose

To transfer control to a labeled point in a loop, a macro, or in the input stream.

Format

LPJUMP *x*

Parameter

x	specifies a label of up to four characters.

Description

A label (denoted by *x*) may contain any ASCII character except blank or comma, but lowercase characters are treated as equal to the corresponding uppercase characters. Transfers within a loop or a macro may be backward or forward, but transfers within the input stream may only be forward. The specified transfer point is a line whose format is one of the following:

```
:x
@EDIT :x
```

where the colon must appear in column 1 (column 7) and must be immediately followed by the label with no intervening blanks. Such a labeled line may not contain any other Text Editor commands, and has no effect if executed rather than used as a LPJUMP target. The search for a label within a macro or loop is downward to the bottom of the outermost loop and then downward from the top of the currently active loop. If the label cannot be found within the loop, an error message is printed and the loop is terminated. Labels should be unique within any individual macro and within a loop entered from the runstream; this restriction is not absolute, but the search algorithm should be understood thoroughly before using nonunique labels. It is inadvisable to transfer out of the currently active loop. The label on the LPJUMP command may be generated by LPSUB substitution, but it is not permitted to generate any part of a label line itself by use of LPSUB. When LPJUMP is used in the input stream, the name of the jump target will be printed, and input solicitation will indicate that a jump is in progress. The @@X C command may be used to stop a jump as for stopping an LPTST skip.

3.3. LPSUB

Purpose

To replace characters in one or more following command or data lines with variable information.

Format

LPSUB,*k* v_1,s_1 v_2,s_2 ... v_i,s_i

Parameters

k	number of lines to be affected (default value is 1; that is, the next line only).
v_i	one or two characters indicating the value to be substituted.
s_i	one or more parameters denoting the character to be substituted for, and other specifications as specified in Table 3-1.

Table 3-1. LPSUB Specifications

Spec.	Format	Description
I	I,*c*	Input line number.
L	L,*c*	Loop counter for currently executing loop of a nest.
L*n*	L*n* ,*c*	Loop counter for loop at depth *n* of loop nest (0 is outermost loop).
LD	LD,*c*	Loop nesting depth.
N	N,*c*	Line number.
LC	LC,*c*	Column in which last string found by the LOCATE or LC command starts.
CC	CC,*c*	Column in which last string changed by the CHANGE command starts.

Table 3-1. LPSUB Specifications (continued)

Spec.	Format	Description
LG	LG,*c*	Length of current text line in characters (note that blank lines have a length of 1).
V	V,*c*	Value computed by most recent COMP command.
T	T,*c*	Time of day (format *hh:mm:ss*).
D	D,*c*	Date (format *dd mm yy*).
T1	T1,*c*	Time of day (all numeric, format *hhmmss*).
D1	D1,*c*	Date (all numeric, ISO format *yymmdd*).
SP	SP,*c*	Spacing value for current line (1 for nonprint files).
TX	TX,*c,x,y*	Portion of current line of text. The character *c* is replaced in the following lines by the *y* characters beginning at character position *x*. If *y* is omitted, a value of 1 is assumed; if both *x* and *y* are omitted, the entire line is assumed. If *y* has an asterisk (*) suffixed, trailing blanks will be removed from the substituted string. A value of zero for *x* will be treated as 1.
QU	QU,*c,x,y*	Portion of response to most recent REM? command. The *x, c,* and *y* parameters have the same meaning as for TX.
MA	MA,*c,n,x,y*	Parameter submitted on most recent call to the macro specified by the name *n*. The meaning of *x* and *y* is the same as for TX. If *x* is explicitly entered as zero, the string will begin with the delimiter which followed the macro name on the macro call line. That is, the delimiter is treated as the 0 (zero) character of the parameter. If *x* is omitted, it is treated as 1. The asterisk may be used following *y* as for the TX substitution.
U	U,*c*	User-id (as determined by the Text Editor).
R	R,*c*	Run-id (original).
RG	RG,*c*	Generated run-id (guaranteed unique by the EXEC).
SI	SI,*c*	Site-id of input device.
NN	NN,*c*	String whose value is null, LN, IL, or NI, depending on what prefixed the most recent uncompleted LOOP command or macro call. This allows a macro, for example, to be called as LN*xxx* and substitute the characters before printing commands in the body of the macro.
OP	OP,*c,x,y*	Portion of response to the most recent OPR* command. This functions in the same manner as QU does for REM? OPR* echoes the reply to PRINT$ as well.
X*a*	X*a*, C	Value of variable set by COMP, where *a* is void or is any alphabetic character.

Description

The LPSUB command replaces characters in one or more following command lines of a loop with variable information. Each v_i is one or two characters indicating the value to be substituted, and s_i is one or more parameters denoting the character to be substituted for and other specifications as given.

Each s_i indicates the character for which the indicated substitution is to be made, and other parameters for substring selection and macro name specification. The specification *k* indicates how many lines are to be affected by the LPSUB command; if *k* is omitted, 1 is assumed (that is, the next line only). A value of 0 for *k* disables substitution. An LPSUB overrides any previous LPSUB still in effect. All occurrences of each character specified in each s_i on the following *n* images will be replaced.

Example:

This loop, containing the LPSUB command:

```
LOOP 6
LPSUB L,$
I ;L,S$;A0,IMAGE,X7;.                 Note the semicolon used as tab character
@EOF
```

will insert the six lines

```
L,S1            A0,IMAGE,X7         .
L,S2            A0,IMAGE,X7         .
etc.
```

into the file.

Only one substitution of each type may be active at a time. (Each L*n* is considered different for different values of *n*.) This means that you can use MA and QU substitution simultaneously, but it is not possible to substitute parameters from two different macros at the same time.

You can use the LPSUB command outside of loops in the same way and for the same reasons as LPTST. The substitution will affect the specified number of input lines.

The LPSUB counter is not decremented for lines skipped by the LPTST or LPJUMP commands. Therefore, if the number of lines to be substituted is different on different loop iterations, the LPSUB,0 command may be necessary to turn off substitution. If this is not done, there is the possibility that an LPSUB command could modify itself on a subsequent iteration of the loop, producing unexpected and mystifying results.

3.4. LPTST

Purpose

To execute conditional statements in a loop.

Format

```
LPTST,n     condition
```

Parameter

n	is number of statements to skip if the specified condition is true.
condition	is listed in 3-2 and 3-3.

Description

The LPTST command lets you execute conditional statements in a loop. If the condition specified is true, the Text Editor will skip the next *n* statements. If the condition is not true, the Text Editor will execute the next statement in sequence. If you do not specify *n* , the default is 1. The Text Editor skips the next statement in true conditions. If *n* is 0, all the loops of a nest will be terminated, not just the current one.

The CSF command also sets the FIND and NOFIND indicators, depending on whether the request was successful or not (as indicated by the setting of bit 35), respectively. If you do not specify a condition on an LPTST command, the skip (or loop exit) will take place unconditionally.

You can make tests for specified conditions. These conditions are described in Table 3-2.

Table 3-2. LPTST Conditions

Condition	Description
FIND	True if the last FIND, LOCATE, CHANGE, LC, or FC command matched a string, or if the last CSF command received a positive status.
NOFIND	True if FIND condition is false.
NEW	True if the current line is a newly inserted or altered image on this edit. Note that MOVE and DITTO lines are NEW, as are lines modified by INLINE and DOC, even if no changes were made to the line.
OLD	True if NEW is false.
ADD	True if the last image read by the Text Editor came from an @ADD file.
NOTADD	True if ADD is false.

You can also make tests for relational conditions on numeric or string values. Table 3-3 describes the allowed values.

The allowable numeric relational conditions are EQ, NEQ, LSS, LEQ, GEQ, and GTR. They represent equality, inequality, less than, less than or equal, greater than or equal, and greater than, respectively. If you specify EQ or NEQ, an additional clause of the form MOD *m* may be added following *expression*, where *m* is a nonzero integer. This form tests the remainder on division of *expression-value* by *m* for zero or nonzero for EQ and NEQ, respectively.

The format for one of these tests is:

LPTST *value relation expression*

where *value* is described in Table 3-3, *relation* is the allowable numeric relational conditions described above, and *expression* is any permitted expression.

The only operators that are valid for string testing are EQ and NEQ. The value following the operator is a literal string. This literal string may begin with a single quote (') character, in which case the string may include blanks and must end with a single quote character (or by the end of the line). You can include a single quote character within a quoted string by writing two single quotes. If you do not begin the string with a quote, it is terminated by the first blank encountered (or by the end of the command). The literal string may, of course, be computed with the LPSUB command.

You can use the LPTST command outside of a loop, with the result that the specified number of input lines will be skipped. This is most likely to be useful in batch mode editing or in an @ADD file. A message will appear indicating the number of lines to be skipped, and, if in demand mode and not in an @ADD file, the skipped lines will be solicited with SKP* preceding the line number. You can use the break keyin (@@X C) to stop a skip; however, the next line will still be solicited with SKP* even though the line will not be skipped.

Table 3-3. LPTST Values

Value	Type	Format	Description
L	numeric	L	Loop counter for currently active innermost loop.
L*n*	numeric	L*n*	Loop counter for loop nested at depth *n*, where *n* is a single digit 0-9. 0 denotes the outermost loop, and a macro call counts as a loop level.
LD	numeric	LD	Loop nesting depth.
N	numeric	N	Current line number.
I	numeric	I	Current value of input line counter (as referenced by the GI command).
SP	numeric	SP	Line spacing for current line (value of 1 if the file being edited is not a print file).
LC	numeric	LC	Column number in which last string found by LOCATE or LC command starts.
CC	numeric	CC	Column number in which last string changed by CHANGE command starts.
LG	numeric	LG	Length of current text line (note that blank lines have a length of 1).
V	numeric	V	Value computed by most recent COMP command.
X*a*	numeric	X*a*	Value of variable set by COMP command, where *a* is void or is any alphabetic character.

Table 3-3. LPTST Values (continued)

Value	Type	Format	Description
QU	string	QU,*x*,*y*	Value is *y* characters starting at character *x* of the reply to the last REM? command. If *y* is omitted, 1 is assumed. If *x* and *y* are both omitted, the whole string is assumed. A value of zero for *x* will be treated as 1.
TX	string	TX,*x*,*y*	Value is a portion of the current line. The meaning of *x* and *y* is the same as for the QU specification.
MA	string	MA,*n*,*x*,*y*	Value is the parameter from the most recent call to the macro *n* . The meaning of *x* and *y* are as for QU. If *x* is explicitly entered as zero, the string will begin with the delimiter that followed the macro name on the macro call line. An omitted value of *x* is treated as 1.
NN	string	NN	If the most recent uncompleted LOOP or macro call had a prefix (one of LN, IL, or NI), the value is a two-character string corresponding to that prefix (in uppercase). Otherwise, the value is the null string.
OP	string	OP,*x*,*y*	Value is *y* characters starting at character *x* of the reply to the last 0PR* command. Functions in the same manner as "QU" does for REM?

3.5. COMP

Purpose

To compute the value of an integer expression.

Format

```
COMP e
COMP* e
COMP! e
COMP:v e
```

Parameters

e specifies the integer expression.

* prints the computed value in decimal.

! prints the computed value in octal.

:*v* the colon must be followed by variable *v*, which may be any of the 27 possibilities X, XA, XB, ..., XZ.

Description

The computed value is always retained for later use with either the LPSUB (see 3.3) or the LPTST (see 3.4) command V specification. The value computed by the expression will be saved as the value of the designated variable as well as for the V specification. The expression *e* may use the operators +,–,*, and /, as well as parentheses for grouping. The only operands allowed are integers, the variables (X, XA, etc.), and the specifications I, L, N, V, LC, CC, and LG, which have the same meanings as they do for the LPSUB command. Integers used in the expression are interpreted as octal if they have a leading zero (Series 1100 convention); otherwise, they are assumed to be decimal. For example, the numbers 015 and 13 represent the same value.

3.6. XTI

Purpose

To allow a command to be entered at a specific point in a loop.

Format

```
XTI
XTI!
XTI*
```

Description

Use this command in loops. It allows a command to be entered and executed at a specified point in a loop. When the Text Editor encounters the XTI command while executing a loop, it will solicit a single command from you and execute it, and then proceed to execute the next command of the loop. This command can be useful if a number of similar but not identical changes are to be made to lines which are located by some complex sequence of commands. For instance, the locate operations may be done in the loop, with an XTI command used at the point where the change is to be made, allowing you to choose between the CHANGE, INLINE, or REPLACE commands.

Use the form XTI! when you want to enter an unlimited number of commands to be read from the input stream. When you finish entering your commands, enter XTI* to resume executing the loop. Similarly, if you desire to do nothing when an XTI or XTI! is reached, you may enter XTI*.

3.7. LPEND

Purpose

Indicates the end of a nested loop.

Format

```
LPEND
```

Description

The LPEND command ends the current loop iteration and returns control to the statement following the corresponding LOOP command, or, if the iteration count is satisfied, gives control to the statement following the LPEND command. You must use one LPEND for each LOOP command with the exception of the outermost. Do not execute the LPEND command conditionally or enter it via the XTI sequence.

3.8. LPX

Purpose

Sets the remaining iteration count for the current nested loop.

Format

LPX *n*

Parameter

n is the iteration count for the current loop.

Description

The LPX command sets the remaining iteration count for the current nested loop to the value specified by *n* . The default value assumed if *n* is omitted is 1. The count includes the current iteration. A value of 0 will have the same effect as LPT,0 . A value of 1 will cause the current loop to end after the current iteration is completed. Therefore, a loop may be ended with control going to the next outer loop by executing LPX and then a skip to the corresponding LPEND.

Loops that do not stop on passing the end-of-file are identified by negative iteration counts. Therefore, if *n* has a negative value, the loop will be placed in nonstop mode. In particular, the command LPX -2 may be used as the first command in a macro definition to indicate that the macro is not to end if the end-of-file is passed; macros normally will stop if the end-of-file is passed.

3.9. MACRO

Purpose

To define a macro, which is a list of Text Editor commands stored in a file named *n* .

Format

MACRO *n*
MACRO* *n*
MACRO? *n, n, n* , ...,
MACRO?
MACRO??

Parameter

n is the macro name.

Description

For each of the first three forms, *n* denotes the name of a macro. Macro names are unique by the first three characters only. The first character must be alphabetic, but the other characters may be alphanumeric; the dollar sign may also be used. If a macro name duplicates the name of an existing Text Editor command, the definition will be accepted, but you will not be able to call the macro.

The first form specifies the entry of a macro definition. Macro definitions are entered exactly like LOOP definitions (including the use of @EOF and @EOF L), except that they are solicited by MAC* . A macro definition operation will destroy any stored loop, so that LOOP! will be invalid.

The second form specifies the deletion of the definition of the specified macro. It is not necessary to delete a definition before redefining a macro; this is done automatically. This deletes only the in-storage definition; a definition (as an *n* -ED-MACRO element) in a file is unaffected. The third form will list the text of one or more defined macros. The fourth form will list the names of all defined macros, and the fifth form will list the text of all defined macros. (For purposes of listing macros, the set of defined macros contains only those known to the Text Editor; those contained in the program files as elements named *n* -ED-MACRO but which have not been called will not be listed.)

Once you have defined a macro, you can call it by specifying its name as if it were a Text Editor command. The remainder of the command line will be stored as the macro parameter, which may be retrieved by the LPSUB and LPTST commands. Macros may be thought of as loops which have an implied iteration count of 1. Macros may use loops. Also, loops and macros may call macros, subject only to the limitation on loop nesting. Each macro call is an additional loop nesting level.

In addition to direct entry of a macro command followed by the text of the macro, a macro may be defined by implicit reference. If the Text Editor encounters a command which it does not recognize, the input file, TPF$, and ED$PF (if assigned) will be searched for an element whose name is *n* -ED-MACRO, where *n* denotes the command (up to three characters). If such an element is found, its text is loaded as the text of a macro with the name given as the command. The macro is then called. In this case, the @EOF L feature is not available.

This mechanism lets you construct a library of Text Editor macros to perform whatever functions are frequently required. This feature cannot be invoked while a loop or a macro is already active, just as it is impossible to define a macro or enter a loop while executing a loop or macro. Macros defined in this manner which have not been called will not be listed by either the MAC? or MAC?? command; these commands can only list macros that have been stored internally by the Text Editor.

A useful example of a macro is the following "delete until locate" macro:

```
1.  MACRO DUL
2.  STK UP
3.  ON BRIEF
4.  LPT N NEQ 0
5.  +
6.  LOOP 99999
7.  LPS MA,$,DUL,0,100*
8.  L.$
9.  LPT FIND
10. LPJ NOF
11. LPX
12. LPJ LPE
13. :NOF
14. D+
15. :LPE
16. LPE
17. STK DN
18. LPS NN,$
19. $(N)
20. @EOF
```

With this definition, you may then enter DUL ABC and all lines will be deleted until a line containing ABC is reached. An explanation of the macro example follows:

1. Name of macro - DUL.
2. Save the modes.
3. Do not print images unless specifically directed by PRINT, QUICK, etc.
4. If current line number ≠ 0 then skip one line.
5. If current line number is zero, + positions to the next line.
6. A large enough number was given as a parameter to loop so as to include all possible lines in the element or file.
7. MA refers to the parameter used for the most recent call of DUL. This parameter will be substituted for $. "0,100*" indicates 100 characters starting at character position 0 and trailing blanks will be removed from the substituted string.
8. Search current line for parameter (no space between "." and "$").
9. If there was a FIND on the previous LOCATE then skip the next line.
10. Unconditional transfer of control to label NOF.
11. Set the loop count by default to 1. Current loop will be terminated after current iteration.
12. Unconditional transfer of control to label LPE.
13. Statement label.
14. Delete current line and position to the next line.
15. Statement label.
16. Indicates end of the scope of the loop (LOOP 99999).
17. Restore the modes that were saved in 2.
18. Substitute the two-character string if the MACRO call had a prefix (LN, IL, or NI) for the $.
19. If a prefix in 18 was present, then the appropriate line number/number will be printed.
20. Every MACRO terminated by @EOF.

NOTE: *A macro with a void body may be thought of as a string variable. A new value is given to it by specifying the desired value on a call to the macro, and the value may be retrieved with either the LPTST or the LPSUB command.*

Because of the use of the prefixes LN, IL, and NI, a macro name may not begin with any of these pairs of characters. These prefixes may be used when invoking a macro; however, see the LPSUB substitution NN for application.

Saving MACROs in ED$MAC

The Text Editor checks the internal name ED$MAC when it is called, when it terminates, and whenever a MACRO or LOOP is created or deleted. If this file is assigned when the Text Editor is called, it will attempt to load a set of saved macro definitions in internal form from ED$MAC. At the other times mentioned, the Text Editor will save all the known MACRO definitions, the current LOOP definition, and the variables X, XA, XB, ..., XZ in ED$MAC, if it is assigned to the run. Definitions saved in this format can be loaded much more rapidly than by the runstream or by the implicit definition method. By this means, it is possible to achieve continuity of MACRO definitions across several TEXT EDITOR operations, irrespective of the nature of the intervening operations, so long as the assignment of ED$MAC is unchanged. If ED$MAC is not assigned to the run, even if it is an attached name, none of the above actions is taken. In other words, the Text Editor will not create or assign this file; that is your responsibility.

NOTE: *If the Text Editor is in input mode and you enter "@EDIT macro-name," any macro line not beginning with @EDIT will be inserted (unless the macro switches to edit mode). You can prevent this by starting each macro line with @EDIT.*

4. Usage Considerations

4.1. Searching Commands

The Text Editor proceeds sequentially through the text. It is therefore more efficient to perform editing operations in a more or less sequential manner starting at the beginning of the text. Searching commands such as LOCATE and CHANGE require much computation and should be used carefully; column limits may be used to speed the search.

4.2. Interrupts with the Text Editor

There are certain processes within the Text Editor which, if indiscriminately interrupted, can cause the processor to fail. To protect against this, the Text Editor stops only at specified points when it is safe to do so. If you wish to interrupt the Text Editor, depress the break key (or MESSAGE WAITING key) at any time. (This step is necessary only if the Text Editor is printing at the time.) The system will respond with:

```
*OUTPUT INTERRUPT*
```

You should answer with @@X C or @@X CO if you wish to cancel the current command. In the first case, backed-up printout may follow before the interrupt takes place. If for some reason the Text Editor's escape method is not satisfactory, you may enter @@X CIO twice. In this case, the Text Editor will return to edit mode, but integrity is not guaranteed.

The interrupt sequence will also have the following effects:

1. Backed-up commands on the same line as the one interrupted (when MCCHAR is in use) will be ignored.

2. If a LOOP or MACRO command is in operation, it will stop.

3. If a LOOP command is being entered, it will not be executed (but loop entry mode continues until an @EOF is encountered).

4. If an LPTST skip operation is in progress, it stops.

5. If an LPJUMP jump operation is in progress, it stops.

4.3. File Name Caution

Files with names of the form ED$*xx* (where *x* is any character) should be avoided since the Text Editor uses such files internally.

4.4. Integer Expressions Instead of Integers

With the exception of the WAIT command and file/element cycle specifications, it is possible to use an integer expression anywhere an integer is permitted, such as for line numbers, column specifications, etc. The expression must be one which would be acceptable to the COMP command and must in addition contain no embedded blanks.

NOTE: Numbers entered with leading zeros are treated as octal numbers.

As an example, the command

```
    P N-3,N+3
```

would print the seven lines surrounding and including the current line.

If an integer expression is the first thing encountered on a line, it must begin with a sign, left parenthesis, or number in order to be recognized as an implied GO or NEXT command. (That is, L+3 is a call on the LOCATE command, not a GO to the third line after the line whose number is the current loop counter value.) If the expression begins with a sign, a NEXT command is implied; if it begins with a digit or a parenthesis, a GO command is implied.

4.5. Column Limits Immediate Specifications

In addition to the use of the LIMIT command, it is also possible to specify column limits for immediate use of a single command only, overriding the limits specified by default or by the LIMIT command. This applies only to the SEQ, LOCATE, CHANGE, and explicit printing commands, but not to commands that print a line implicitly because of alteration or transfer of file position. The syntax for specification of column-limits may be any of those specified in Table 4-1.

Although the syntax of the immediate column limits specification is common to all commands which accept it, the position in which it must be specified differs. For more detailed information, see 2.2.8 and 2.2.41.

Table 4-1. Immediate Column Limits Syntax

Format	Left Limit	Right Limit
[*n*,*m*]	*n*	*m*
[*m*]	1	*m*
[*n*,]	*n*	132
[,*m*]	1	*m*
[,]	1	132
[ ]	1	132

NOTE: If the Text Editor is locally reconfigured for 160 character lines, the values of 132 in the above table should be changed to 160.

4.6. Default for F, FC, L, LC, and C Commands

For each of the F, FC, L, LC, and C commands, the previous target specification is saved and retrieved if the command is entered without any specification. F, FIND, and FC reference one saved string; L, LOCATE, and LC another; and C and CHANGE a third. For the C (and CHANGE) command, the saved string contains the number of lines, global (G), REP, and ALL specifications (if any), as well as the old and new strings. For the F and L type commands, the saved string contains only the search argument, and different restrictions (such as number of lines scanned, column limits, or number of occurrences) may be used each time.

The commands may be entered in the form F?, L?, and C? . Each of these will display the current default value to be used for the associated command. Other specifications on the command will be ignored. If no command of the associated type has been given, nothing will be displayed.

4.7. LN, IL, and NI Feature

You can use the characters LN, IL, or NI to prefix any command. If this is done, any lines of the file printed by the command will be prefixed with the associated line number, input cycle line number, or both, respectively. None of the three prefixes is considered to be part of the command for purposes of abbreviation (that is, LNDITTO may be abbreviated to LNDIT but not to LND, since LND would be an abbreviation for LNDELETE). There are two special cases: LNS is an abbreviation for LNSITE (although "S" is not a valid command), and LN (rest of line blank) will print out the current and input line numbers and remain in edit mode. LN*n* , LN+*n* , LN-*n* , IL*n* , IL+*n* , IL-*n* , NI*n* , NI+*n* , and NI-*n* , where *n* is an integer (expression) are all permitted. If LN, IL, or NI is used to precede the name of a macro, the macro will be called, and the LN, IL, or NI will be available as the LPSUB parameter NN. These pairs of letters cannot be used to begin the name of a macro.

4.8. Names for ASCII Control Characters

There are a number of commands which accept a single character as a parameter (TAB, CCHAR, LCHAR, MSCHAR, TCHAR, TCCHAR, and SHCHAR). For any of these, it is permissible to specify the name of an ASCII control character (such as ACK, BEL, DC3, DEL) instead of typing the character itself. Either BL or SP may be used to stand for the blank. The use of a name is also permissible instead of the octal code for the EXCH command. The STATUS command will type the name instead of the actual character if a control character is in use as a special character for the Text Editor; this avoids adverse effects on the terminal in use, such as activating a paper tape reader.

4.9. Print File Operations

The Text Editor performs certain special actions when editing a print file. These actions include:

- The new line created by a CHANGE, INLINE, or RETYPE command will be given the spacing count of the replaced line, rather than a spacing count of 1.
- Lines transferred by the DITTO and MOVE commands will retain their previous spacing counts. Print control images included in the range to be transferred will be moved as well.
- If the ADD command is used to add a print file to a print file, the lines added will retain their spacing counts, and any print control images will also be added.
- If a new file (not element) is written by the SPLIT command, it will be created as a print file with the appropriate label information, spacing counts will be preserved, and print control images will be written. The resulting file can be printed with @SYM just as if it had been created by the symbiont complex.
- Print files created by the Text Editor (whether by processor call specification [as in @ED,U] or by the SPLIT command will have Fieldata/ASCII information in each image and each 224-word block will begin with an SDF image control word. These files will be compatible with all symbiont operations.

4.10. Edit Mode Commands in Input Mode

The Text Editor allows any editing command to be submitted while in input mode. The format is:

@EDIT *command*

where *command* denotes any command which is valid in edit mode. The letters EDIT must be in columns 2 through 5, and the command must begin in column 7. If the command is omitted, the Text Editor will return to edit mode. This feature is convenient for setting up tab characters, AUTO counts, and so on when inserting a new element using the I option.

This format is acceptable in edit mode as well as in input mode, but it is superfluous. When the LOOP command is used and all or part of the loop is to be executed in input mode, @EDIT format may be used for any commands which are to be executed in input mode, such as LPSUB or LPTST.

This feature may be used when typing ahead (as in @@CQUE mode) to guarantee being in a particular mode. To force operation to edit mode, enter @EDIT; to force operation to input mode, enter @EDIT INPUT. The desired mode will be entered regardless of which mode was previously active.

If the MCCHAR command is being used in input mode to indicate multiple lines on a single input line, each line which is to be interpreted as a command must have its own @EDIT following the MCCHAR character; without it, the command would be treated as input.

4.11. Character Command Processing

The commands CCHAR, MCCHAR, and SHCHAR define characters for which special action is to be taken on input. These characters are processed before the command on a line is analyzed, and then the character designated is deleted from the line. Thus, if the same character is specified twice in succession on any of these commands, the second time the effect will not be what is intended. Since the character is deleted from the line, the command will appear to have no specification, which will deactivate the feature, and, if the MCCHAR character is involved, a blank line will appear to be present, switching the mode (unless in EOF mode) of the Text Editor.

4.12. Reusability

The Text Editor is reusable. Successive uses of the Text Editor will not require an initial program load, which will conserve system resources. Note, however, that this means that the system log contains fewer entries for the Text Editor than the number of actual distinct edits performed.

One consequence of the interaction between reusability and the command feature in input mode is that if a demand user attempts to terminate the Text Editor with a transparent control statement (@MSG, @LOG, etc.), the Text Editor sign-off message will be delayed until a nontransparent control statement (including @ED) is entered.

Reusability may be prevented by terminating the Text Editor with the @ENDX statement.

4.13. Restrictions and Limitations

The following restrictions and limitations apply:

1. If a FORTRAN data file is updated by the Text Editor, the links used to hold backspacing information will be lost. Hence, do not use the FORTRAN BACKSPACE statement on an updated FORTRAN file. Use of the R option with the Text Editor will avoid accidental modification of the file.

2. Due to the internal structure of the Text Editor, it is not possible to edit print files containing lines with spacing counts in excess of 63. Such lines will appear to be deleted.

3. The GI command may not be used when editing print files, since the necessary information is not available, due to the presence of print spacing information.

4. Any command that references line numbers can only reference line numbers that are less than or equal to 262,143.

5. When you insert ASCII control characters into the text and request your output file in Fieldata, the ASCII control characters will be converted to question marks since there are more ASCII characters than Fieldata characters.

6. The device name on the SITE and LNSITE commands have special requirements. If the device name begins with an X, or if it begins with an L, N, I, V, LC, CC, or LG followed by a nonalphabetic character, the commands will not work unless you specify all parameters (that is, the SITE *num1 num2 device* format).

4.14. The ED$TC File

The Text Editor attaches the internal name ED$TC to a file whose name is *project-id* *ED$TC*run-id*. For demand users, this file will normally be a cataloged file; if another run with the same original run-id is active and using the Text Editor or for a batch run, this file will be a temporary file. ED$TC (if cataloged and not temporary) is assigned with the D option. This allows the implementation of the AUTO RECOVERY feature. If a run terminates normally, ED$TC will be deleted, and there will be no auto recovery. If a (demand) run terminates abnormally, such as by system crash, line drop, terminal timeout, @@TERM statement, or operator keyin (SM *site-id* T), the ED$TC file will be retained. This permits auto recovery when a new run is initiated and the Text Editor is executed.

NOTE: The new run must have the same project-id and run-id as the run which terminated abnormally.

If you fail to start such a run and use the Text Editor in it, the ED$TC file will remain cataloged indefinitely. Therefore, you should be aware of this situation and delete a file which is no longer required.

The internal structure of the ED$TC file is of no importance to most; however, since it contains certain link addresses which depend on the internal structure of the Text Editor, the data in ED$TC must be verified to be correct. This is done through the use of a validity constant (VALCON). The value of VALCON will be different for different levels of the Text Editor, so ED$TC cannot be used between levels of the Text Editor. VALCON may also be used to detect possible corruption of ED$TC due to hardware errors, and a change in this value is also possible if the Text Editor itself has a software failure. If a VALCON error occurs, it is necessary to erase ED$TC before calling the Text Editor again.

4.15. Obsolete Commands

A number of commands are still defined in the Text Editor which are now considered obsolete and are thus not documented in detail. These are listed in Table 4-2.

Table 4-2. Obsolete Text Editor Commands

Command	Description
BOT B	Same as APPEND.
BRIEF BR	Same as ON B.
CLIMIT	Same as LIMIT C.
END	Same as OMIT.
GO *n*	Goes to line *n* of file, where *n* is an expression.
HEAD H	Same as 1 (goes to first line of file).
NEXT *n* N *n*	Moves forward *n* lines in the file (backward if *n* is negative). If *n* is omitted, 1 is assumed.
O *n*	Same as P+*n*, if QUICK mode is OFF. Same as Q+*n* if QUICK mode is ON.
PLIMIT PL	Same as LIMIT P.
SAVE	Same as LIMIT C.
TOP T	Same as 0 (goes to top of file).
VERIFY V	Same as OFF B.

5. Error Messages

AT __ JUMP HISTORY: __
Jump history given after 'IOPR' or 'IGDM'.

ATTEMPT TO DIVIDE BY 0
Self-explanatory.

ATTEMPT TO EDIT PROGRAM FILE AS A DATA FILE
Data attempt on program file.

AUTO COMMAND CANNOT BE USED — AUTO FILE IN USE, ROLLED OUT, OR DISABLED
AUTO cannot be done.

AUTO FILE BAD, NONEXISTENT, OR SAVED IN READ-ONLY MODE
AUTO file not assigned or not available.

AUTO FILE COULD NOT BE ASSIGNED

AUTO FILE FROM DIFFERENT VERSION OF ED - ENTER @ERS ED$TC.
Validity constant from different version of ED.

AUTO NOT ALLOWED IN BATCH

AUTO RECOVERY

BAD TAB

CASE UPPER ASSUMED

CHANGES IGNORED!
No changes in READ ONLY mode (use UP to allow changes).

CHARACTERS TO BE SUBSTITUTED GREATER THAN MAX LINE LENGTH
String length greater than maximum configured line length.

COLUMN REFERENCE ERROR
Column limits syntax error (1st limit > 2nd limit, 2nd limit = 0, or 2nd limit exceeds LN132)

CYCLE NUMBER OUT OF RANGE
Cycle does not exist.

DEPRESS PUNCH ON
Signal for you to push "ON" button on the paper tape punch.

DISABLED — ACCEPTED
File is disabled but is accepted.

ELEMENT __ NOT IN SPECIFIED FILE __
Element could not be located in the given file.

ELEMENT DESIGNATION ERROR
Error in element designation.

EMODE __ AT __
Error contingency.

END ED. NO CORRECTIONS APPLIED
Element of file not updated.

FILE *filename* CANNOT BE ASSIGNED EXCLUSIVELY
File is assigned to another run.

FILE DESIGNATION ERROR
Error in file designation.

FILE EXPANSION FAC REJECT
File could not be expanded.

FILE FAC REJECT: __
File facility reject.

FILE IN FIELD __ CANNOT BE ASSIGNED
File cannot be assigned.

FILE KEYS CANNOT BE RECOVERED BY AUTO — OMIT, @FREE, RE-@ASG YOUR FILE WITH KEY(S), AND DO @ED,A...

FILE NOT PROGRAM FILE FORMAT
Not a program file.

FILE REASSIGNED 10% LARGER
File size increased by 10 percent.

HAS NO EQUIPMENT ASSIGNED TO IT
No equipment code on an ASG.

ILLEGAL COMMAND AFTER LAST
Six commands are illegal after LAST.

ILLEGAL COMMAND WHEN FILE IS NOT PRINT FILE
SSP can be used only with print files.

ILLEGAL COMMAND WHEN USED AT LINE 0
Certain commands are not permitted at line 0.

ILLEGAL MOVE - DITTO ASSUMED
MOVE line number range included current line.

IN USE BY ANOTHER RUN
File used by another run.

INCORRECT MODE SPECIFIED
Incorrect mode name for ON/OFF.

INPUT FILE READ LOCK-ED MUST READ FROM OUTPUT FILE BEFORE WRITING TO IT
Need READ key, or file is write-only.

INPUT MODE ASSUMED
Self explanatory.

INVALID CHAR: __
Invalid character.

INVALID COMMAND WHEN EDITING A PRINT FILE
Input file line number invalid for print file.

INVALID EDIT CHAR: __
Invalid character for INLINE editing.

INVALID SYMBIONT NAME - ENTER NAME:
Give proper symbiont name.

INVALID TYPE
Invalid processor-mnemonic or octal value given for TYPE.

I/O ERROR
Input/Output error.

JUMP TARGET NOT FOUND
LPJUMP - Label not found.

JUMPING TO LABEL __
Jump to given label. (Occurs with LPJUMP in runstream.)

LINE NUMBER EXCEEDED
Maximum line number is 262,143

LINE REFERENCE ERROR
Wrong line number - possibly beyond end-of-file.

LOOP COMMAND ILLEGAL IN XTI SEQUENCE
LOOP not allowed in XTI sequence.

LOOP/MACRO STORAGE SPACE EXCEEDED
Too many loops and/or macros. Use LOOP+ to expand storage space.

MACRO CANNOT BE DEFINED OR DELETED DURING LOOP OR MACRO EXECUTION
LOOP or MACRO active - cannot define or delete.

MAIL FILE FULL
No room left in mail file.

MAIL FILE ROLLED OUT OR DISABLED
Mail file not available. Enable file and/or wait for file to roll in before entering MAIL? again.

MAXIMUM LOOP NESTING DEPTH EXCEEDED
Too many loops nested.

MINIMUM LINES ALLOWED:
Minimum for AUTO.

MISSING DELIMITER: __
Missing string delimiter.

MISSING NUMERIC
Number expected - syntax error.

MISSING OPERAND
OPERAND expected - syntax error in COMPUTE command.

MISSING OPERATOR
OPERATOR expected - syntax error in COMPUTE command.

MISSING TERMINATION CHAR
Missing terminator.

NEGATIVE NUMBER NOT ALLOWED
Illegal negative number.

NO ADDRESSEE FOR MAIL
Specify user-id on MAIL command.

NO CHANGES SINCE LAST TOP
UNDO rejected - no changes since last TOP.

NO CORRECTIONS APPLIED
Element or file not updated.

NO FILE SPECIFIED, TPF$ ASSUMED
No file on processor call line.

NO FIND
String not found.

NO LOOP STORED
LOOP! but there is no loop.

NO SDF HEADER WORD IN FILE
File not standard SDF format.

NONEXISTENT CYCLE
Invalid cycle.

NOT ON FASTRAND FORMAT DEVICE
Not sector format mass storage.

OUTPUT FILE OVERFLOW
Output causes file size to be exceeded.

OUTPUT FILE OVERFLOW—FILE CANNOT BE EXPANDED
ED was unable to enlarge the file.

OUTPUT FILE WILL OVERFLOW, AND CANNOT BE EXPANDED DUE TO ONE OR MORE OF THE FOLLOWING @ASG OPTIONS (D, K, R)

OUTPUT FILE WRITE LOCK
Need WRITE key, or file is read-only.

OVERFLOW OF STORAGE FOR LOOPS, MACROS VALUE OF LOOPRES
LOOPRES exceeded - maximum storage for loop and macros.

PROCESSOR CALL STATEMENT IS IN ERROR
Bad processor call syntax.

PROGRAM FILE TABLES FULL — ELEMENT NOT SAVED
Overflow of file.

PROGRAM FILE UNDEFINED
Undefined file or element.

READ-ONLY MODE CHANGES IGNORED!
A command which changes the file (including an attempt to enter input mode) was entered in read-only mode.

READ-ONLY MODE - SPLIT LINES NOT DELETED
In READ-ONLY mode split lines are not deleted.

REQUESTED LINE DELETED OR ALTERED
Line number used on GI command has been affected by previous editing operations.

SEQUENCE NUMBER LIMIT EXCEEDED
An edit command attempted to access a line number that is larger than 262,143.

SYNTAX ERROR BEFORE COLUMN __
Examine ED command preceding the designated column number.

TOO MANY LEFT PARENTHESES
Self explanatory.

TOO MANY RIGHT PARENTHESES
Self explanatory.

TOO MANY SPECIFICATIONS
INFOR buffer overflow.

UNABLE TO PERFORM UNDO ON CHANGES SINCE LAST TOP

U OPTION ASSUMED FOR DATAFILE
Data file - assume U option.

USE A OPTION TO RECOVER AND THEN SPLIT TO DIFFERENT FILE
A option for AUTO RECOVERY needed to recover element or file when temporary file has overflowed.

WARNING: CYCLE LIMIT WAS ZERO: NEW LIMIT = 5
Cycle limit should never be zero.

WARNING: I, U, C, OR UP NOT PERMITTED WHEN FILE IN USE
READ ONLY mode if file in use and Y option on processor call.

WARNING: NEW ELEMENT WILL REPLACE EXISTING ELEMENT
Element already exists but will be replaced by element created by editing operation.

WARNING: NEW FILE WILL REPLACE EXISTING FILE
File already exists but will be replaced by file created by editing operation.

WARNING: ON EXIT OUTPUT FILE WILL BECOME A DATA FILE
Output file will be a data file, but is currently a program file - potential for data loss exists.

WARNING: TEMPORARY FILE ASSIGNED INSTEAD OF FILE AT TIME OF AUTO

WARNING: STATUS OF FILE NOT SAME AS AT TIME OF AUTO

IS AN ILLEGAL COMMAND
Illegal command or non-existent macro was referenced.

Index

USER COMMENT SHEET

Comments concerning the content, style, and usefulness of this manual may be made in the space provided below. Please fill in the requested information.

This User Comment Sheet will not normally lead to a reply to the originator. Requests for copies of manuals, lists of manuals, pricing information, etc. must be made through your Series 1100 site manager, to your Sperry Univac representative, or to the Sperry Univac office serving your locality. Software problems should be submitted on a Software User Report (SUR) form UD1–745. Questions of a technical nature regarding either the manual or the software should be submitted on a Technical Question (question/answer) form UD1–1195. These forms are available through your Sperry Univac representative.

Customer Name: ______________________________ System Type: ____________________

Title of Manual: ____

UP No.: ________________ Revision No.: __________ Update: ____________________

Name of User: ______________________________________ Date: ________________

Address of User: ____

Comments: Give page and paragraph reference where appropriate.

Please rate this manual.	Good	Adequate	Not Adequate
Organization of the text	______	______	______
Clarity of the text	______	______	______
Adequacy of coverage	______	______	______
Examples	______	______	______
Cross references	______	______	______
Tables	______	______	______
Illustrations	______	______	______
Index	______	______	______
Appearance	______	______	______

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

ATTN.: SOFTWARE SYSTEMS PUBLICATIONS

P.O. BOX 64942
ST. PAUL, MINNESOTA 55164

FOLD