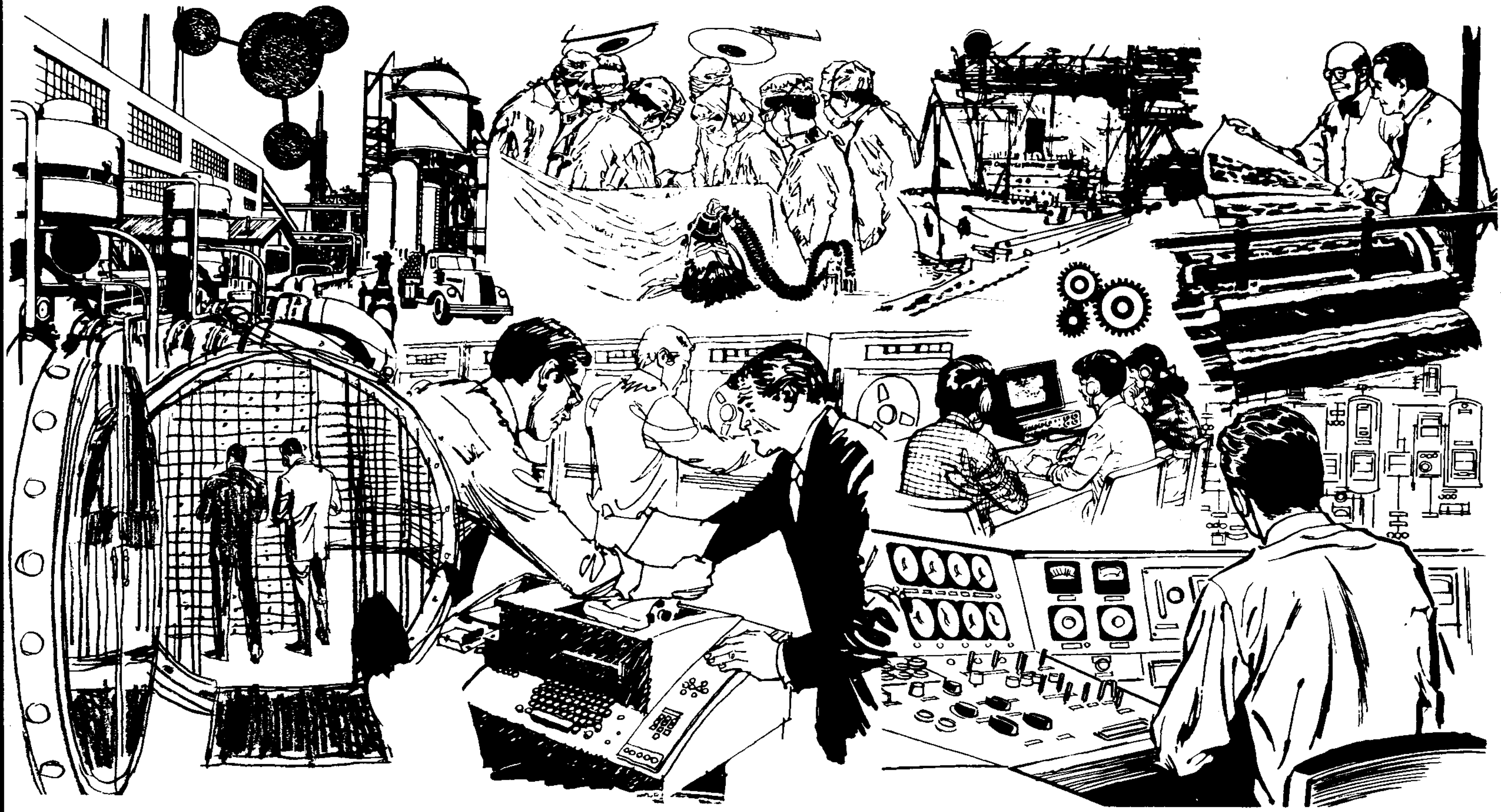
BEM:EDT — OS/3
Interactive Editor

User Reference



SPERRY UNIVAC

UA-0141 Rev. 3

PAGE STATUS SUMMARY

ISSUE: Update A to UA-0141 Rev. 3

Section	Page Number	Update Level
Cover		Orig.
Title	*i*	Orig.
Copyright	*ii*	A
Preface	*iii,iv*	Orig.
Contents	*v,vi*	Orig.
1	1	Orig.
2	1,2	Orig.
	3	A
	4-8	Orig.
3	1-24	Orig.
4	1-13	Orig.
5	1	Orig.
6	1-16	Orig.
A	1-3	Orig.
B	1	Orig.
C	1-27	Orig.
D	1-8	Orig.

Section	Page Number	Update Level

Section	Page Number	Update Level

All technical changes are denoted by an arrow (⟶) in the margin. A downward arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward arrow (↑) is found. A horizontal arrow (⟶) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

BEM:EDT — OS/3
Interactive Editor

User Reference

SPERRY UNIVAC

UA-0141 Rev. 3
Update A

SPERRY UNIVAC NEWSCOMP Newspaper Composition Program was used by Application Services in typesetting this publication.

A User Comment Sheet is provided at the back of this publication for your comments. If the sheet has been removed, comments may be mailed to Sperry Univac, Attn: Manager, Application Services, P.O. Box 500, Blue Bell, PA. 19422.

PREFACE

This manual describes the SPERRY UNIVAC Basic Editor Monitor (BEM) Interactive Editor (EDT) which is designed to enable the conversational user to create new files or update and/or display existing files. The manual contains six sections:

- Section 1 — Introduction to EDT — presents the purpose and functions of the Editor and its salient features.
- Section 2 — Editor Concepts — describes the notation used in the command formats, programming notes applicable to all commands and to the Editor in general. In addition it also details the procedures required to log on, invoke the Editor, edit a file, terminate a session, and log off.
- Section 3 — Editor Commands — presents detailed descriptions and examples of the commands.
- Section 4 — Editor Procedure File Commands — contains descriptions of those commands associated with Procedure files.
- Section 5 — Error Detection and Recovery — briefly describes the different errors detected by the Editor and relates how the Editor attempts to recover from the errors or otherwise responds.
- Section 6 — BEM operation — details elements of the BEM monitor, and how to use it.

Appendixes contain the following information:

- Appendix A — Alphabetic Listing of EDT Commands — A complete alphabetical listing by command of all the Editor and Procedure file commands. The function of each command, its functional category, and the page in the manual where the detailed description is located, are provided.
- Appendix B — Operand Definitions — An alphabetic listing of all operands used by the Editor commands with a detailed definition of each operand.
- Appendix C — Error Messages — An alphabetic listing of all BEM error messages. The reason for each message is given as well as the procedure to recover from the error when possible.
- Appendix D — Sample EDT Session — Various examples of using the Editor commands, the Procedure files, and a sample Editor session.

This manual is one in a series pertaining to BEM. The others are:

BEM — OS/3 Basic Editor Monitor, User Reference UA-0139
BEM: BASIC —OS/3, User Reference UA-0140
BEM: RSP —OS/3 Remote Spoolout Processor, User Reference UA-0243

Readers of this manual should also become acquainted with the *SPERRY UNIVAC OS/3 System Service Programs,* UP-8062.

CONTENTS

1 INTRODUCTION TO EDT

The SPERRY UNIVAC Basic Editor Monitor (BEM) Interactive Editor (EDT) provides extensive facilities for creating and updating text elements in OS/3 library files. The updating facilities include inserting, copying, saving, deleting, and concatenating elements. Editing of existing elements is accomplished in a work area independent of the OS/3 Library files. Elements consist of text or Editor commands. Entire elements are created that contain all commands or a mixture of commands and text. These elements, when in Editor Procedure files, are executed during an Editor session.

Text and commands are entered conversationally at the user terminal. Up to 128 characters can be entered on a single terminal line. The Editor accepts a single line starting with a start-of-entry (SOE) and ending with the cursor followed by the user depressing the TRANSMIT key. A number is assigned by the Editor to each line of text or command entered at the terminal. Each time a line of text is entered, the Editor automatically increments the line number. The increment is preset to 1, but it can be changed as desired. The line number is not incremented for commands.

Editor commands are provided which allow manipulation of the line numbers. Manipulation includes resetting the line number and accessing specific lines as desired.

The Editor can cause whole lines or an entire element to be printed prior to or after editing. Since all printing is done at the user's terminal, there are Editor commands to specify the maximum number of lines to be displayed at one time.

Editor functions are summarized in Appendix A. The user should become familiar with Appendix A and the programming notes in Section 3. This background will provide the material required to use the Editor to its fullest potential.

Due to the variety of commands, and the many ways in which these commands may be combined, certain functions may not be clear to a user who is not acquainted with interactive editing. This user should read this manual completely through once, then go back and review individual commands. The new user is also directed to Section 2.1, which provides examples of complete commands and their functions.

2 EDITOR CONCEPTS

The Editor is an interactive program which runs under the monitor BEM. All user commands are entered through a UNISCOPE terminal. The user should be familiar with the *UNISCOPE Display Terminal, Operating Reference Manual,* UP-7788.

The OS/3 Editor combines the functions of the Librarian with those of a text editor. The Editor can read and write source, copy, and procedure files written in Library format. Elements may be read by the Editor, corrected, and rewritten to the Library. In addition, new elements may be created. With the Editor many of the Librarian functions may be executed interactively, reducing the possibility of losing an entire element due to keypunch error. Corrections may be verified as they are entered to further reduce the chances of error. The user may look at a line before modifying to check that the line numbers on the listing are valid.

The Editor allows selected words in the element to be changed by entering a single command.

All changes made by the Editor are made to a work file, not to a library file directly. Consequently mistakes are easily corrected without rebuilding a file. The Editor has commands to transfer portions or all of the Library elements to and from its work area. Once in the work area, lines from the element may be displayed at the terminal and modified by the user.

As the Editor reads a Library element into its work area, line numbers are assigned to each card image. These numbers are included when lines are displayed at the terminal. Editor commands use the line numbers to identify the portion of the work that is to be processed by the command.

2.1 FOR THE NEW USER

After the terminal has been connected to the system by dialing the computer the user must log on to the BEM system. First depress the TRANSMIT key to get the system's attention. BEM responds with:

```
PLEASE LOGON
▷/
```

The user must log on by entering the command:

/LOGON *id*

where *id* is the user's identification code.

To execute the Editor, enter:

```
/EXEC EDT
```

The Editor will display its ready message:

```
UNIVAC OS/3 EDITOR READY (VER 5.0)
```

followed by the current line number (1.0).

The Editor is ready for the user. To enter new data lines, type the lines on the terminal one at a time. After typing each line, depress the TRANSMIT key, which transmits to the Editor all the characters between the last start-of-entry, (▷), and the cursor. The Editor accepts the line and stores it in the user's work file on disk. The user need not be aware of his work file, as it is managed entirely by the Editor.

The current line number defaults to 1.0 when the work file is empty. When the work file contains text, the current line number is set to the next line after the end of the last text line, but may be redesignated by the user via an Edit function.

The Editor always displays the current line number and awaits the input line from the user. Since commands to the Editor are distinguished by an *at* sign (@) as the first character, a data line cannot start with an at sign. To allow data lines which start with at signs, the Editor interprets a double at sign (@@) in columns 1 and 2 as a data line starting with a single at sign.

In order to see what is entered the user may request the Editor to display all or some of the work file. The EDT PRINT command displays lines at the terminal. For example:

```
@ PRINT
```

displays the entire work file. Selected lines may also be specified, as:

```
@ PRINT 10-15
```

displays the 6 lines numbered from 10 to 15.

To change certain parts of entered lines the CHANGE command may be used. This command allows the user to change specific words in any or all of the lines. To correct the spelling of the word 'ERRRER' in the first five lines entered, the following command would be used:

```
@ON 1-5 CHANGE ALL 'ERRRER' TO 'ERROR'
```

The ON clause specifies the line range to be searched; if omitted the entire range is searched. The ALL parameter instructs the Editor to change every occurrence of the word in each line. If all is omitted, only the first occurrence of the word in each line is changed. To check to see that the changes were correctly made, the user can display every line that contains the word 'ERROR' with the following command:

```
@PRINT 'ERROR'
```

To see only the first line containing the word, specify the FIRST parameter:

```
@PRINT FIRST 'ERROR'
```

Now to eliminate several lines which are wrong the user may use the DELETE command.

```
@DELETE 4-6,8
```

This command removes the four lines numbered 4, 5, 6, and 8 from the work file. The DELETE command can also be used to remove lines containing a specific word:

```
@DELETE 'ERRRER'
```

If the DELETE command is used with no operands, it will erase the entire work file.

```
@DELETE
```

When the data is completely entered in the Work file it may be written to a permanent OS/3 Library file with the WRITE command.

```
@WRITE DATA123,PERMFILE,PACK01,S
```

This command specifies the element name of the information being written, the file name into which it is to be written, and the name of the disk pack which contains the file and the source type designation. The element is written in OS/3 Library format and may be accessed later by both the Editor and the OS/3 Librarian (LIBS). To get the same or a different element back into the Editor work file for editing, the READ command is used.

```
@READ SOURCE2, COBOLIB, PACK18,S
```

NOTE: The editor cannot access or write to a cataloged file that contains periods and/or qualifiers in its filename. ←

The element is read into the Editor's work file beginning at the current line number and may be edited.

The current line number is updated to reflect the first free line after the newly read lines. For example, if the line number was 1 before the READ statement and 48 lines were read in, the current line number would be reset to 49.

To duplicate lines of code, the COPY command may be used.

```
@COPY 1-10 TO 101
```

This command will copy line 1 to 101, line 2 to 102, line 3 to 103, and so on up to line 10 which will be copied to line 110.

To change the current line number so that lines may be entered at a different location, the user uses the following format:

```
@ 100.5
```

This will set the current line number to 100.5. The next line entered will be stored at line 100.5. The second line entered will be stored at 100.6. The current line pointer increment is determined by the number of digits specified to the right of the decimal point. If the command was entered as:

```
@100.500
```

the line number would still be set to 100.5, but the increment would be set to .001.

With the COPY command, the increment used for the output is taken from the TO clause of the COPY command. Thus the command sequence:

```
@50
@COPY 1-10 TO 101.5
```

would cause the lines 1 to 10 to be copied to lines 101.5, 101.6, 101.7, . . . , 102.4 because the line number specified in the COPY command specified 101.5. This number does not affect the increment used for entering commands.

If a set of lines have been entered at one location, and are to be moved to another location, the MOVE command is used.

```
@MOVE 1000-1100 TO 242
```

The MOVE command works like the COPY command except that the original lines are deleted after the move. After the above command is executed, there will be no lines stored at locations 1000 through 1100. The lines that were there originally will be stored at locations 242 through 342.

Lines can be referenced by a line number, and also by a character string within the lines. Thus any lines with a given search string can be copied or moved to a specified line range.

```
@ON 1-1000 COPY 'SEARCH' TO 2000
```

The above command creates a copy of all lines containing the word 'SEARCH' starting at line 2000.

When entering data lines it is often convenient to define tab stops for maintaining column formats. To set tabs and define a tab character, the SET command is used:

```
@SET CHAR=;,TABS=10,16,40,72
```

The above command defines the tab character to be a semicolon (;) and the tab stops to be at columns 10, 16, 40, and 72. Tab advancement is always in the forward direction. Thus if the first tab character is entered at column 11 data will be advanced to column 16, not back to column 10.

Often it is necessary to make minor modifications to many lines but the CHANGE command cannot be conveniently used. For this purpose use the UPDATE command. UPDATE displays selected lines to the user one at a time to be corrected and transmitted to the Editor.

```
@UPDATE 10-40
```

This command allows the user to update lines 10 through 40. Each line in the range will be displayed, one at a time, at the terminal. The user can then make any corrections to the line on the screen and retransmit it. The corrected line will be stored in place of the old line and the next line in the range displayed at the terminal. The command will terminate when the end of the range is reached, or when the user enters a command instead of the updated line.

Often when specifying a search string, only certain occurrences are of interest. If it is known that the string always starts in certain columns, a column range may be specified.

```
@ON 10-30 COLUMN 1-9 PRINT 'LABEL'
```

will print only those lines which contain the word 'LABEL' starting in columns 1 through 9. If the word starts in any other column the line is not displayed. The COLUMN parameter may be specified in any command containing a search string. In certain other commands its function differs. If a CHANGE command specifies a change string but no search string, the change is always made at the specified column regardless of what was in those columns.

```
@ON 10-20 COLUMN 40 CHANGE TO 'COMMENT'
```

will overlay the data starting at column 40 with the word COMMENT. Data before column 40 and after 46 will be left unchanged. Similarly the INSERT command will insert a character string between existing characters in the specified lines.

```
@ON 10-20 COLUMN 40 INSERT 'COMMENT'
```

will cause the word COMMENT to be inserted between the data at columns 39 and 40. No existing data will be changed but the new line will be longer due to the inserted characters.

At certain times the user will want to communicate with the system operator. Any user message may be transmitted to the operator with the TYPE command.

```
@TYPE THIS IS A MESSAGE
```

will display 'THIS IS A MESSAGE' on the OS/3 operator's console. It will be prefixed with a BEM message number and the user's terminal ID.

To terminate an Editor session, the user should first save his work file. Then to return to system mode the HALT command must be entered:

```
@HALT
```

To free the terminal and edit work areas, log off as follows:

```
/LOGOFF
```

2.2 COMPARISON WITH VS/9 EDT

The OS/3 BEM/EDT is patterned after the VS/9 Editor. (See VS/9 EDT Reference Manual EA-038-2-00.) Many of the simpler commands are identical in format. This section outlines the major differences of BEM/EDT and VS/9 EDT.

Commands in BEM/EDT are free-form; however, the formats to which VS/9 users are accustomed will be accepted. The following list of commands is accepted with no modification:

```
@ON line-range PRINT [FIRST] search-string
@ON line-range CHANGE [FIRST] [ALL] search-string TO change-string
@PRINT line-range
@DELETE line-range
@COPY line-range TO copy-to location
@MOVE line-range TO copy-to location
@COLUMN column-range INSERT change-string
@COLUMN column-range CHANGE TO change-string
@HALT
```

When used with a search-string the DELETE command is slightly different since it always deletes lines as opposed to character strings.

BEM/EDT also has an UPDATE command which is not included in VS/9. The main difference, however, is that BEM/EDT commands are free-form. Various commands may be combined in many ways, whereas VS/9 has fixed formats.

A column range is specified differently. With BEM/EDT the user must specify a COLUMN parameter rather than including the column range in the line range.

The READ and WRITE commands are slightly different to account for the difference in file structure between VS/9 and OS/3.

@READ *elt, filename,vsn,type*

@WRITE *elt, filename,vsn,type*

Several functions are handled differently. For instance tabs are set with the @SET command.

```
@SET TABS=10,16,40,71,CHAR=;
```

BEM/EDT has some commands that are not included in VS/9. The @TYPE command transmits a message to the OS/3 operator. The @HELP command gives a detailed description of the last error.

2.3 COMPARISON WITH OS/3 LIBRARIAN

Many of the Editor functions duplicate interactively certain OS/3 Librarian (LIBS) functions. (See *OS/3 System Service Programs,* UP-8062 Part 2.)

To create a new element with the Librarian requires keypunching the source deck and then using LIBS. The Librarian functions ELE and EOD must be used:

```
// JOB CREATE
// DVC 20 // LFD PRNTR
// DVC 50 // VOL PACK // LBL FILENAME // LFD FILE1
// EXEC LIBS
/$
   FIL  D1=FILE1
   ELE  D1,S,EXAMPLE
     (source deck)
   EOD
/&
// FIN
```

To use EDT instead, the element would be entered through the terminal instead of being keypunched. Then after it was correctly stored in the Editor's work area it could be added to the file with:

```
@WRITE EXAMPLE,FILENAME,PACK,S
```

To make corrections with LIBS requires the above job control and the control statement:

```
COR    D0,S,EXAMPLE
(correction cards)
EOD
```

With EDT the element must first be read with the READ command:

```
@READ EXAMPLE,FILENAME,PACK,S
```

The full power of the Editor is then available to the user for making corrections. Entire lines may be reentered or only selected words throughout the file may be changed. At any time, however, the user may display the lines which are about to be corrected to make sure they are the right lines. Corrected lines may also be displayed to ensure that the changes are made correctly. None of the changes are made permanent until the work area is written out; thus, at any time the user may start over with the uncorrected element.

To remove lines with LIBS requires the SKI command, with the sequence numbers of the lines to be removed:

```
SKI    EX120900                                      EX110300
```

With the Editor, lines are deleted by their Editor line numbers:

```
@DELETE 330-420
```

The lines may be displayed before they are deleted by using the PRINT command. In this manner the wrong lines are less likely to be removed.

The Librarian allows the current location to be set with the REC command:

```
REC     (sequence number)
```

To set the current line number with the Editor use:

```
@(line-number)
```

To add sequence numbers with LIBS the SEQ command is used:

```
SEQ     DO,S,EXAMPLE,73,EX000000,100
```

To do the same thing with EDT use:

```
@SEQ    'EX000000'BY 100 COL 73
```

This sequences the element in the work file.

Although the Editor is not designed for file manipulation certain functions can be executed if necessary. To add elements from one file to another with LIBS would require:

```
// JOB ADD
// DVC 20 // LFD PRNTR
// DVC 50 // VOL DISC1 // LBL FILE1 // LFD LIB1
// DVC 51 // VOL DISC2 // LBL FILE2 // LFD LIB2
// EXEC LIBS
/$
     FIL DO=LIB1,D1=LIB2
     ADD DO,M,MACROS,D1
/*
/&
// FIN
```

Doing the same thing with EDT requires:

```
@READ   MACROS,FILE1,DISC1,M
@WRITE  MACROS,FILE2,DISC2,M
```

Certain functions are left exclusively to the Librarian. These include:

Copying files (COP DO,,,D1)
Directory listings (COP DO)
Renaming elements (REN)
Packing files (PAC)

3 EDITOR COMMANDS

Editor commands are divided into two subsets: directives and data manipulation commands. Directives provide the interface among the user, the Editor, and the OS/3 System, and do not affect the work file. The data manipulation commands operate on the data in the work area.

Commands are identified by a single *at* sign (@) as the first character in the line. Lines containing other first characters or double at signs (@@) are considered to be data lines.

All commands and keywords may be abbreviated. Most require only the first character; none more than the first three letters. The minimum abbreviation is indicated by the underlined portion of the command in Appendix A.

3.1 DATA MANIPULATION COMMANDS

The data manipulation commands are free-form commands built up of operations, line, and column range specifications. The operations describe the functions to be performed on the work area; the specifications limit the operations to selected portions of the work area.

All commands must start with an *at* sign (@). Except for setting the current line pointer. The second non-blank character in each command must be alphabetic; not numeric or another *at* sign. Otherwise, the operations and specifications of data manipulation commands may be included in any order within the line..Spaces are treated as delimiters, but in most cases they are not needed. Command words are terminated by spaces, numerics, or punctuation. If commands are formatted carefully, delimiters will be present automatically in most cases. Only rarely will spaces need to be added for delimiting purposes.

All keywords used in commands may be abbreviated if the user does not wish to enter the complete command word. Most keywords may be abbreviated by entering at least their first character; for example, the PRINT command may be abbreviated using P, PR, PRI, or PRIN. There are certain commands where a single letter abbreviation may lead to confusion, for example COLUMN, COPY, and CHANGE. In these instances two- or three-letter abbreviations should be used. Section 3.3 contains the minimum abbreviations which the Editor will recognize.

3.1.1 Line and Column Range Specifications

The line and column range specifications define the portion of the work area that is to be affected by the operations. The user may select specific lines, specific columns in lines, or lines containing a specific character string. Any or all of the various specifications may be included in any data manipulation command.

3.1.1.1 ALL

This specifies that changes and insertions are to be repeated for the entire column range in each line. If omitted, changes are made only once in each line in the range.

```
@ON 1 CHANGE ALL '   ' TO '*'
```

will change every space in line 1 to asterisks.

```
@ON 1,2 CHANGE '   ' TO '*'
```

will change only the first space in lines 1 and 2 to asterisks.

3.1.1.2 BY (increment)

This clause specifies the increment to be used with the SEQUENCE command.

It must be used if, and only if, the SEQUENCE command is used. The increment may be specified as a one- to five-digit decimal number. Any number in the range 1-99999 is valid.

```
@SEQ 'EDT00000' COL 73 BY 200
```

If the number specified contains more digits than there are numeric places in the sequence-string, the addition will be performed without modifying the alpha characters. For more examples and a complete description see the SEQUENCE command description.

3.1.1.3 COLUMN (column-range)

COLUMN specifies either the column range in each line that is to be searched for, the search string, or the column where a change or insertion is to take place.

```
@ON & PRINT 'MOVE'
```

will print all lines containing the word 'MOVE' and

```
@ON & COLUMN 11-20 PRINT 'MOVE'
```

will print all lines containing the word 'MOVE' starting in columns 11 to 20.

The column range refers to the column in which the search string is to begin. If, in the above example, the word was contained in columns 19 to 22, it would be printed. However, if the word was in columns 10 to 13, it will not be printed. If the column specification is omitted, the entire line is searched.

The percent symbol (%) can be used to identify the starting column of the last search-string found. This symbol will be set by any search command and will point to the first occurrence of the string in the last line found.

```
@ FIND      'SEARCH'
@ COL   %   ON   ?      CHANGE TO 'NEW'
```

The FIND command will locate the first occurrence of the search-string and the second command makes a change to that line at the column at which the search-string began.

```
@ON 10 CHANGE ALL TO '*'
```

will cause the entire line to become asterisks. However,

```
@ON 10 CHANGE ALL COLUMN 1-10, 40-70 TO '*'
```

will put asterisks only in columns 1-10 and 40-70 and leave the rest of the line as it was.

3.1.1.4 (file-parameters)

This clause specifies the OS/3 Library file to be accessed by a READ or WRITE command. It defines the element, the file containing the element, and the disk pack on which the file resides. The clause must be entered immediately following the READ or WRITE command.

File parameters specify the element name, element type, file name and password, and volume name. If the file has been catalogued with a password, the user must supply the read/write password to gain access to the file. A disk volume name, if catalogued with the file, need not be used in the command. The element type may be source, PROC, or macro and is designated by an S, P, or M, respectively. If type is omitted, the element is assumed to be source. Only the element name need be specified if there is a default file for the user's account.

All entries used for file parameters must be alphanumeric, with the first character alphabetic. The element name may be up to 8 characters, the file name up to 44 characters. The password may be up to 6 characters, and the volume name 6 characters. The general file-parameter specification appears as follows:

element, filename [(*password*)], [*volume*], [*type*]

For example:

```
@READ OPR,MACROLIB(SYST),OS3RES,M
@WRITE ELEMENTZ,LIBOUT,,S
@READ ELTNAMEG,,,P
```

3.1.1.5 FIRST

This specifies that the operation is to terminate after the first line is encountered which contains the search string.

```
@PRINT 'FORMAT'
```

will print every line in the work area which contains the word FORMAT.

```
@PRINT FIRST 'FORMAT'
```

will print only the lowest numbered line that contains the word FORMAT.

3.1.1.6 TO (change-string)

Indicates the operand of a CHANGE command. This must be specified whenever the change operation is included. The string which follows may be either a character string or a hexadecimal string. A character string must be enclosed in apostrophes; a hexadecimal string must be preceded by an X. If apostrophes (') or quotes (") are to be included in a character string they must be repeated. That is, to indicate IT'S, the string must be entered as 'IT' 'S'. A hexadecimal string must contain only valid hex characters 0-9, A-F.

3.1.1.7 TO (copy-to location)

This specifies the output location and increment for a COPY or MOVE operation. It may be specified as a single line number or a series of line numbers. Certain restrictions pertaining to overlapping ranges are noted under the COPY operation. This clause must be included whenever a COPY or MOVE operation is included. If, during a COPY or MOVE operation, the output line number becomes 9999.0000 then the increment is changed to .0001 automatically.

3.1.1.8 (search-string)

The search-string specifies a string which must be found in each line to be processed by the data manipulation command. Either a character string or a hexadecimal string may be specified. Both forms have the option of selecting any occurrence of the string or only those bounded by delimiters. Delimiters recognized by EDT are

.(+ !/*); - /, ?:'=' and space.

Quotes (") are used when delimiting is required, apostrophes (') when it is not. Quotes and apostrophes may be used in the same string to indicate leading or trailing delimiters only.

A character string consists of any string of characters enclosed in apostrophes or quotes. If quotes or apostrophes are to be included in the string, they must be repeated. Thus, ' " " QUOTE" " ' would be the string "QUOTE". If the symbol # is to appear in a search string it must be typed as ## to distinguish it from an EDT variable (See Section 4.2).

A hexadecimal string consists of hex digits (0-9, A-F) enclosed in apostrophes or quotes and preceded by an X. For example,

X'123AF'

The following table summarizes the options for a search string:

Search-string	Found in line:/ABCDE,F,G?	Reason
'ABC"	no	'C' char not followed by delimiter
"ABC'	yes	Delimiter not required after 'C'
'ABC'	yes	In searched string
"ABC"	no	Not bounded by delimiters
"F"	yes	Bounded by delimiters
X"C6"	yes	Represents hexadecimal 'F'
'Z'	no	Not in searched string
'F'	yes	In searched string

3.1.1.9 ON (line-range)
(line-range)

Specifies the lines that are to be searched within the work area. The ON form is required if the line range is to be specified as the first parameter in a line. Elsewhere the keyword ON is optional. A line-range consists of combinations of the line numbers of the form:

```
A
A-B
A,B
A-B,C-D
&
*
?
```

or any combination of the above separated with commas.

An asterisk (*) indicates the current line number. A question mark (?) indicates the last line found by a search command. It is set by any command containing a search string and by INSERT, CHANGE, and SEQUENCE commands; all other commands leave it unchanged.

An ampersand (&) indicates the entire line range, A-B indicates an inclusive range, and A indicates a single line. A,B,C,D are line numbers of the form:

dddd.dddd

Each digit is a decimal digit (0-9). A maximum of 4 digits may be specified on each side of the decimal point. Zero (0.0) is an invalid line number; all others in the range 0.0001-9999.9999 are valid.

Of the following line ranges, those in the left column are valid; those in the center column are invalid.

Valid	Invalid	Reason
643.1049	19401.344	Too many digits to left of decimal
1215.34	12.98365	Too many digits to right of decimal
12.3-12.9	14.34-13.1	Decreasing range
1-2,3-5,8	1-2,3-5-8	3-5-8 invalid specification
1-99,40-55	1-23,0-2	0 is invalid line number
&	$	$ is not a valid line number

3.1.1.10 NOT

The inclusion of this parameter reverses the sense of any search string test. That is, if a line would have been processed without NOT, it will not be processed when NOT is included.

For example:

```
@PRINT NOT 'FORMAT'
```

will print all lines which do not contain the word FORMAT.

3.1.1.11 DESEQ

This clause can only be used on a READ command. It causes all lines read from the library file to be written to a work space location specified in the line itself; the line number is taken from the columns specified by the column range. The number itself can consist of 1-8 numeric digits. If the number is zero or greater than 9999 9999, the line is written sequentially after the previous line written as if DESEQ had not been specified. If less than eight digits are found, zeros are padded on the left to make eight digits The decimal point is always placed so that there are four digits to either side of it.

3.1.2 Operations

The operations may be combined in almost any manner. Certain combinations are disallowed due to the inherent conflict of the operations. These restrictions prohibit combining READ and WRITE commands with each other or with operations using the terminal. Other restrictions preclude duplication or contradiction. For instance, COPY and MOVE cannot both be specified, nor can INSERT and CHANGE. For a complete description see Table 3-1.

Table 3-1 Invalid Combinations of Operations

Operations	COPY	CHANGE	DELETE	INSERT	MOVE	PRINT	READ	UPDATE	WRITE	SEQUENCE	NUMBER
COPY	no				no						
CHANGE		no		no						no	no
DELETE			no		no		no				no
INSERT		no		no						no	no
MOVE	no		no		no						no
PRINT						no	no		no		
READ			no			no	no	no	no		no
UPDATE							no	no	no		
WRITE						no	no	no	no		no
SEQUENCE		no		no						no	no
NUMBER		no	no	no	no		no		no	no	no

3.1.2.1 CHANGE

The CHANGE command in conjunction with the TO (change-string) clause, instructs the Editor to replace all occurrences of the given search-string or a specific location with the specified change string.

```
@ON 1-50 CHANGE 'WRONG' TO 'RIGHT'
```

will replace the first occurrence of WRONG in each line in the range 1 to 50 with RIGHT.

```
@CHANGE ALL "ERROR" TO' '
```

will remove all occurrences of the word ERROR which are preceded by and followed by delimiters. All lines in the work area are searched and the lines are collapsed to fill in the vacated space.

```
@ON 100-105 CHANGE COL 10 TO 'TITLE'
```

will cause the word TITLE to replace the existing characters starting at column 10 on lines 100 to 105.

```
@CHANGE 10,15 ALL COLUMN 1-72 TO '*'
```

will generate a row of asterisks extending from column 1 to 72 in lines 10 and 15.

```
@COPY 1-10 TO 20 CHANGE 'ALPHA' TO 'BETA'
```

will copy lines 1-10 to lines 20-29 changing the first occurrence of ALPHA to BETA in each line. The lines at 1-10 are not modified by this instruction. Only those lines which contain the string 'ALPHA' are copied.

3.1.2.2 COPY

The COPY command instructs the Editor to duplicate a range of lines at a location, specified by the TO (copy-to location) clause. The word COPY indicates the operation to be performed and the TO clause specifies the output location of the operation.

```
@COPY 1-10 TO 25
```

will create a copy of lines 1-10 at lines 25-34.

The COPY command will also cause a single line to be propagated through a range of lines by specifying overlapping copy-to location and line-range.

```
@COPY 110-120 TO 111
```

will produce 10 copies of line 110 at lines 111-120.

The line increment used for the output of this function is that specified by the TO clause. If the TO clause is 10.0 with an increment of 0.1 then the copied lines will be stored at locations 10.0, 10.1, 10.2, etc.

If the command

```
@COPY & TO 2
```

is attempted, a loop will result since the lines which are copied will later be recopied before the end of the range can be reached. This loop will continue until the work area is filled (several thousand lines) or until the user interrupts and returns to monitor mode.

The COPY command may also be used to collect all lines containing a specific search-string.

```
@COPY 'SEARCH' TO 1
@ON 100-500 COPY 'SEARCH' TO 1
```

Multiple output locations may be specified as

```
@COPY 1-10 TO 20,30
```

If the output locations do not overlap the input line range, this form of the command is identical to two consecutive commands as follows:

```
@COPY     1-10   TO    20
@COPY     1-10   TO    30
```

While this command may be used without specifying a line range as in the first example above, this is not recommended, since it will often result in a loop. This loop depends on the exact contents of the work area. If the first few lines contain the search string or if lines before the output location contain the search string then a loop will result. However, if there are not enough lines containing the search string for the output location to catch up to the input location, the command used will not loop. (For other examples of the COPY command see the CHANGE command.)

3.1.2.3 DELETE

The DELETE command removes specified lines from the Work file.

```
@DELETE 1-10
```

removes lines numbered from 1 to 10 from the work area.

```
@ON 10-30 DELETE 'BAD LINE'
```

removes all lines in the range 10 to 30 that contain the characters BAD LINE.

```
@D
```

removes all lines in the work file. This also illustrates an example of keyword abbreviations; a D is used to reference the DELETE command.

If the user desires to delete the entire work space to begin editing a new element, the command

```
@DROP
```

should be used.

3.1.2.4 INSERT (change-string)

The specified string is inserted in the text line at the column specified by the COLUMN parameter. Existing characters are shifted to insert the string without losing existing characters.

```
@ON 2-4 COLUMN 5 INSERT '123'
```

will insert the characters 123 between the characters currently at columns 4 and 5.

```
@INSERT '*'
```

will insert an asterisk in front of every line in the work area.

```
@ALL COL 5,10,15 ON 100-200 INSE 'FIVES'
```

will insert FIVES before columns 5, 10, and 15 on the specified line-range. The insertion is made preceding the columns which were originally numbered 5, 10, and 15. The column numbering does not change after the first insertion takes place so that 10, 15 refers to the columns 10 and 15 as they were before the command was initiated.

The ALL parameter allows multiple insertions to be made in a single line.

COL and INSE are used as abbreviations for COLUMN and INSERT.

3.1.2.5 MOVE

The MOVE command functions like a COPY command followed by a DELETE command on each line. It will not loop because the lines are deleted after being copied, but before the next line is copied. Like the COPY operation the MOVE also requires the use of the TO parameter.

```
@MOVE 1-10 TO 50
```

will produce a copy of lines 1-10 at lines 50-59 and the lines at 1-10 will be deleted.

If multiple output locations are used with the MOVE command, only the first location is normally used, because the input lines are deleted after the first move.

```
@MOVE 1-10 TO 20,30
```

In the preceding command, lines 1-10 are copied to 20 and deleted, thus no lines are copied to 30. However, the following example shows an instance where multiple outputs can be used:

```
@MOVE 1-10 TO 0.5,20
```

In this case lines 1-10 are moved to 0.5 through 1.4, then lines 1 through 1.4 are moved to 20. The result is that the lines originally numbered 1,2,3, . . . , 10 are numbered 0.5,0.6, . . . , 0.9,20,21, . . . ,24, respectively.

3.1.2.6 PRINT

The PRINT command displays selected lines from the work area at the terminal.

```
@PRINT 1-10
```

displays lines 1-10.

```
@P 'ABC'
```

displays all lines containing the characters 'ABC'.

The PRINT command may be used with other commands to validate changes after they are made.

```
@ON 100-200 CHANGE "ZETA" TO 'UPSILON' PRINT
```

will make the changes in the line range and display the resultant lines after each change. This also allows the user to terminate the command after some number of changes has been made by using the @SET PAGE=*mm* feature of the PRINT command.

```
@UPDATE 1-100 PRINT
```

will allow the user to update each line in the range and, after updating, the corrected line will be displayed at the terminal. With this command each line is displayed twice; once to allow the user to update it and again after the updates are made.

After a predefined number of lines have been printed to the terminal (see @SET command), the user will be asked to request the next screen when he is ready for it. Two replies are allowed from the following message:

```
CONTINUE (Y OR N)?▷
```

A reply of N will end the command and allow the user to enter the next command or data line. A reply of Y will display another screen of lines, and possibly another CONTINUE query.

The CONTINUE message is not displayed if the editor is run in batch mode, or if the page size is set to 255.

3.1.2.7 READ

The READ command transfers an OS/3 Library file to the user's work area. Lines are read in starting at the current line number and incremented by the current increment. It requires the specification of the file parameters after the keyword READ.

```
@READ    DATA1,WORKFILE,USER01
```

transfers the element named DATA1 in a file named WORKFILE on the disk pack USER01.

```
@READ    DATA1,WORKFILE,USER01    'OLD'
```

will transfer only those lines containing the word OLD.

It is the user's responsibility to inform the operator that certain packs need to be mounted and to ensure that the appropriate job control was filed to allow user access to that pack. The Editor will inform the user if an element, a disk file, or a disk volume could not be located. Catalogued files may be referenced without the volume number; the password associated with the file must be entered if the file was catalogued with a password. The form of the command for a catalogued file with password is:

```
@READ    NEWPROC,PROCLIB(RDPASS),,P
```

The volume number has been omitted, as it can be found in the catalog, but the commas are required to ensure the PROC(P) designator is in the correct position. The parameter RDPASS in parentheses is the read password which matches the one catalogued for the file PROCLIB. Allowable files are S, P, and M for source, PROC, and macro respectively. If none are specified S is assumed.

Because line numbers are not defined until after an element is read by EDT, the use of a line-range is not permitted with the READ command.

3.1.2.8 SEQUENCE (sequence-string)

The SEQUENCE command causes the specified string to be stored at the column specified in the line range. The string is incremented by the increment specified with the BY clause for each new line in which it is inserted. This command is needed to sequence source decks which are to be updated with the OS/3 Librarian. The following command would be used to sequence an assembly program:

```
@COLUMN    73 SEQUENCE 'EDTO0000' BY 200
```

In every line of the file this will store the sequence number starting in column 73. The initial sequence value will be EDT00000 incremented by 200 each line. If the column parameter is omitted the sequence number will be stored in column 1. Thus the following command may be used for COBOL programs which expect sequence numbers in columns 1 to 6:

```
@SEQ    '001000' BY 100
```

The SEQUENCE and BY parameters must always be specified together. However the command may be combined with any other commands except CHANGE and INSERT.

```
@WRITE SOURCE,FILE2 SEQ    'SS000000'COL 73 BY 200
```

The command will write contents of the work area into FILE2 with an element name of SOURCE, sequencing the lines as they are written. The sequence numbers will be in columns 73 to 80, and will start with SS000000 being incremented by 200 with each new line.

When a line range is specified only the lines in that range will be sequenced. If a column range is specified only the starting column is used unless the ALL parameter is included. In that case the sequence number will be placed in the line at all locations in the column range. If a search string is specified only that string will be replaced by the sequence number. If the ALL parameter is specified with the search string the same sequence number will be used for each change in a line. That is, the sequence string is not incremented until a line is finished.

```
@ON    1-100 PRINT 'LABEL' SEQUENCE 'NAME01' BY 1
```

With this command, the first occurrence of LABEL will be replaced by NAME01, the second by NAME02, and so forth. Only lines in the range 1 to 100 will be searched and thus lines which have changes made will be displayed at the user's terminal.

The sequence string must have its rightmost characters numeric (0-9) to be incremented. Only the numeric characters on the right end of the string are incremented. All characters to the left of the rightmost alphabetic character (and that character) are not changed by the increment.

A second form of sequence string consists of an asterisk:

```
@COLUMN 73 SEQUENCE *
```

When this form is used the sequence string consists of eight numeric digits formed from the line number at each line (with decimal point removed). This format can be used with the DESEQ option on the READ command to preserve EDT line numbers in an OS/3 Library file.

For example, if the work space contains the following lines:

```
1.0000      ONE
1.5000      TWO
11.7530     THREE
20.0000     FOUR
```

then the result of the command "@SEQUENCE * COLUMN 10" will be:

```
1.0000      ONE       00010000
1.5000      TWO       00015000
11.7530     THREE     00117530
20.0000     FOUR      00200000
```

3.1.2.9 UPDATE

The UPDATE command displays a selected portion of the work file, one line at a time at the user's terminal. The UNISCOPE cursor is positioned at the end of the line for easy retransmission. The user may change any or all of the line before retransmitting it.

```
@ON 1-10 UPDATE 'BAD'
```

will display for updating only those lines in the range 1 to 10 which contain the word BAD. When used in conjunction with other commands UPDATE provides the user with a chance to override changes before they are stored in the work file.

```
@CHANGE 'ONCE' TO 'TWICE' UPDATE
```

will format the changes of ONCE to TWICE in each line and then display the formatted line to the user. He may transmit the line to store it with the changes made, or he may make other changes before retransmitting. The line with changes due to the CHANGE command and due to user updating will be written to the work file. The UPDATE command may be terminated at any time by transmitting a command instead of the updated line. If the line being updated contains an at sign (@) in column one, it will be interpreted as a command when transmitted back. It is therefore the user's responsibility to add the second at sign needed to distinguish a data line. Tabs are not processed on updated lines so that the current tab characters may be entered with the UPDATE command.

Note that it is also possible to update lines without the UPDATE command by setting the line number and then transmitting the update line to be stored at the line number.

```
@10.5
(insertion line)
```

This line will be stored at 10.5.

During an UPDATE command, the user has three additional options. Instead of transmitting a data line, or terminating the UPDATE with an EDT command, one of the following three commands may be used:

@STRIKE	This command deletes the line being updated and then continues processing the UPDATE command.
@NOCHANGE	This command causes the update line to be left as it was prior to the execution of the command.
@CONTINUE	This command causes the UPDATE command to continue with the UPDATE option removed.

The above options and the UPDATE command combined with a CHANGE command, give the user the ability to verify changes as they are made. For example, consider the following command:

```
@ON 10-500 CHANGE "OPEN" TO 'CLOSE' UPDATE
```

On each line processed by this command the user has the option of deleting the resultant line (@STRIKE), not allowing the change to be made (@NOCHANGE), or letting the CHANGE command finish without the UPDATE option (@CONTINUE).

Note that the STRIKE and NOCHANGE options cannot be used when the UPDATE command is combined with a COPY command. If this combination is needed, the COPY command must be executed first, then the UPDATE command. The CONTINUE option, however, may always be used.

3.1.2.10 WRITE

This command transfers selected lines from the work area to an OS/3 Library file specified by the file parameters. The file parameters must follow the WRITE keyword.

```
@WRITE DATA01, WORKFILE, USER01   1-150 'ONLY'
```

will transfer only those lines in the range 1 to 150 containing the word ONLY. (See READ command for the definition of the file parameters.)

If the element (DATA01) already exists in the file the user will be warned that the element is about to be overwritten and given a chance to abort the command. If the element does not exist in the file it will be created. The file (WORKFILE) must exist on the specified pack and the pack (USER01) must be mounted.

The Library directory is not updated until the entire element is written to the file. Thus, errors encountered during the transfer will not affect the status of the file for subsequent I/O. See Appendix C for a list of errors.

As with the READ command, a password is required if it is in the catalogue, and a file type may be specified:

```
@WRITE COPYELT, COPYLIB(WRTPAS),,M
```

3.1.2.11 FIND

This command locates the first line in the file having the specified characteristics. This command can be used to set the ? line number without printing any lines.

```
@ON 1-30 FIND 'SEARCH' COL 40-70
```

If the first occurrence of the characters 'SEARCH' in columns 40 to 70 had been found on line 25.1 then ? would be set to a value of 25.1.

3.1.2.12 NUMBER (sequence-string)

This command causes sequence numbering to be inserted in subsequent lines typed at the terminal. The BY clause specifies the numbering increment; the sequence string specifies the initial value. The column clause may be used to specify the column in which the sequence string is to begin; if omitted, the numbering will be inserted before column 1. The NUMBER command remains in effect until the next EDT command is entered.

```
@ NUMBER '1000'    BY 100
```

would start numbering at 1000 and increment by 100 all data lines entered from the terminal. The tabs are processed before the numbers are inserted.

3.1.3 EDT Processing Description

All EDT data manipulation commands are built on a general command processing loop. Each operation described in Section 3.1.2 specifies the type of operation to be performed. The line and column range specifications described in Section 3.1.1 define the portion of the work area to process.

The general processing loop is shown in Figure 3-1.

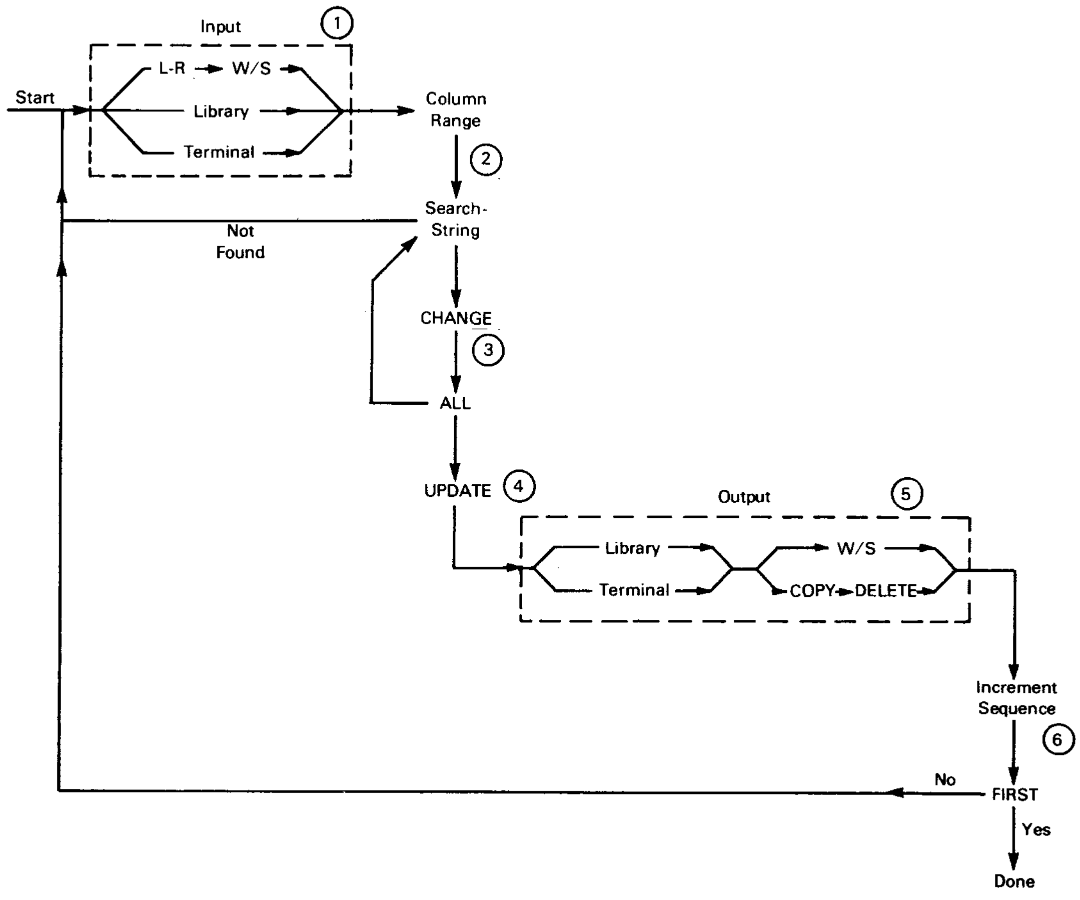


Figure 3-1 EDT Processing Loop

① Input may come from any of three sources. Lines may be read from the user's terminal, the work space, or from an OS/3 library file. The actual input source used depends upon the operation specified: READ implies source from a library file, NUMBER from the terminal, and most other operations from the work space. When input is from the work space, only lines in the specified line range are read.

② If a search-string is specified, it must be found in the line if the line is to be processed. If the column range is omitted the search-string may occur anywhere in the line, but if a column range is specified the search-string must start in the specified range.

③ If a change string is specified (as a result of CHANGE, INSERT, SEQUENCE, or NUMBER), changes are made now. If the ALL option is specified, repeated changes are made, but if ALL is omitted only one change is made to a line.

④ The line with changes made is now displayed for update if the UPDATE option was specified. If a command is returned on the update reply, the loop may be terminated here or the conditions of the loop processing may be modified.

⑤ Output may go to one or more locations. If WRITE was specified, output will go to an OS/3 library file. PRINT will cause the lines to be displayed upon the terminal. In general, changed lines will be written back to the work space. However, if COPY is specified, the changed lines will only be written to the location specified by the COPY-TO option. The DELETE option causes the lines to be deleted from the input location of the work space and suppresses rewriting of changed lines. As indicated in Figure 3-1, it is not possible to display lines on the terminal if output is to a library file, or to write lines back to a copy location and the original work space location.

⑥ At the end of one cycle through the loop, the sequence string is incremented to its next value and the FIRST option is tested. If specified, the loop terminates and EOF processing occurs. Otherwise, the loop is reentered to process the next data line.

Table 3-1 shows the options that are specified for each command keyword. Keywords specify the input and output locations and the ranges and strings that are permitted in the command.

Table 3-1 Keyword Command Options

Command	A	B	C	D	E	F	G	H	I	J	K	L	M	N
COPY	●				●				●		●	●	●	
CHANGE	●				●				●	●		●	●	
DELETE	●			●					●			●	●	
FIND	●								●			●	●	
MOVE	●			●	●				●		●	●	●	
PRINT	●						●		●			●	●	
UPDATE	●				●			●	●			●	●	
READ		●			●				●				●	
WRITE	●					●			●			●	●	
SEQUENCE	●				●				●	●		●	●	●
INSERT	●				●				●	●		●	●	
(DATA)			●		●									
NUMBER			●		●				●	●			●	●

A—Source is work file
B—Source is library file
C—Source is terminal
D—Source is to be deleted
E—Output is work file
F—Output is library file
G—Output is terminal
H—Update command
I—Search string allowed
J—Change string allowed
K—Copy-to location allowed
L—Line range allowed
M—Column range allowed
N—By increment allowed

3.2 DIRECTIVES

The EDT directives are a set of commands which instruct the Editor to modify its parameters or to interact with BEM in some way. None of the directives involve the work area or permanent files in any way.

3.2.1 HALT

The HALT command terminates the Editor and returns the user to monitor mode. All EDT work areas are released. (See also 3.2.9.)

```
@HALT
@HA
```

3.2.2 HELP

The HELP command provides a more detailed explanation of the last error encountered by the user. This command should be used immediately after the error message is displayed, since in some instances the error may be lost by the execution of other commands.

```
@HELP
@H
```

3.2.3 SET

The SET command defines various parameters to the Editor.

```
@SET [PAGE= mm, LINE=ll, CHAR=xx, TABS=n1, n2, n3, n4, n5, n6, n7, n8]
```

The PAGE keyword defines the maximum number of lines to be displayed at a time on the user's UNISCOPE. It should be specified as one line less than the actual screen size to allow room for the CONTINUE query. When the user requests the Editor to display lines, it stops after displaying *mm* lines as defined by the @SET PAGE=*mm* command. The user may continue to have lines displayed or terminate the command by responding to the message:

```
CONTINUE? ▷(Y OR N)
```

which is displayed when the screen is filled.

Permissible values for *mm* are 1 to 255. The larger values are useful when a hard copy terminal is being used. If PAGE=255, the CONTINUE message is not displayed.

The LINE keyword defines the maximum number of data characters to be displayed on a UNISCOPE line. The value should be 10 characters less than the screen size to allow for the line number and SOE characters. For example,

```
@SET PAGE=11, LINE=70
```

would be used for a 12 × 80 UNISCOPE and

```
@S P=15,L=54
```

would be used for a 16 × 64 UNISCOPE.

The CHAR keyword defines the character to be recognized as a tab character in lines accepted by the Editor from the terminal.

The TAB keyword defines the columns to be used as tab stops when a tab character is found. A maximum of eight columns may be specified with the TAB keyword. If the column list is omitted, previously set tabs will be cleared.

```
@SET CHAR=;,TAB=10,16,40,72
```

would be used when entering assembler source statements and

```
@S C=#,T=7,72
```

could be used for entering FORTRAN source statements.

Tab advancement is always in the forward direction. If the FORTRAN tabs were in effect when the following lines were entered:

```
1 2 3 4 5 6 7#X=1.0
1 2 3 #A=B*C
#WRITE(2,10)
4 5 6#C=15.4#SEQNUMB
#D=5#SEQ#NUMB
```

It would be stored in the work area as:

```
|1      |7                                  |72
|-------+-----------------------------------+----
|123456 |7                                  |X=1.0
|123    |A=B*C                              |
|       |WRITE(2,10)                        |
|456    |C=15.4                             |SEQNUMB
|       |D=5                                |SEQ#NUMB
```

If the tab character is entered beyond a tab position, it will cause a skip to the next tab stop. If the number of tab characters entered exceeds the number of tab stops defined, the extra tab characters are treated as ordinary characters.

Previously set tab stops may be cleared by entering the following command:

```
@SET TAB=
```

If no parameters are entered, the current settings are displayed.

For example,

```
@SET
```

will display:

```
CASE=UPPER   PAGE=23   LINE=128   TAB CHAR=;   TABS=10 16 40
```

3.2.4 @(line-number)

This command sets the current line number and increment to be used for data entry. The next data line entered from the terminal will be stored at the current line number and the line number will be increased by the current increment. The increment is specified implicitly by the number of decimal places entered in the line number.

```
@100
```

sets the line number to 100 and the increment to 1.

```
@100.00
```

sets the line number to 100 and the increment to 0.01.

A data line may be entered in the same statement by entering a colon (:) immediately following the line number. The data is stored at the new line number.

```
@10:This line is stored at line 10
@101.21:This line is stored at 101.21
```

Examples:

(line-number)	Set to	Increment
10.	10.	1.
10.	10.	1.
12.	12.	1.
12.10	12.1	.01
12.1	12.1	.1
0.123	.123	.001

If the current line number is incremented to 9999.0000 by EDT, the current increment will be automatically changed to 0.0001. This is to prevent the user from creating lines with a line number greater than 9999.9999.

Instead of a line number, a plus (+) or minus (−) sign may be specified. These indicate that the new line will be calculated by adding or subtracting the current increment and current line number. A data line may follow the plus or minus sign if a colon is the first character after the sign.

Examples:

```
@+
@−: data line.
```

3.2.5 TYPE

The TYPE command transmits a message from the user to the OS/3 System operator.

```
@TYPE PLEASE MOUNT DISK PACK 'USER01'
```

would display

```
VI12 TRM1 PLEASE MOUNT DISK PACK 'USER01'
```

on the operator's console. VI12 is the system message identification number and TRM1 is the user's terminal ID.

NOTE: When in LOWER mode, messages must be entered in all upper case, because lower case will not be displayed on the 90/30 console.

3.2.6 LOWER

The LOWER command instructs EDT to recognize upper and lower case input. Prior to the execution of this command, data entered is translated to all upper case characters. When in lower mode, command keywords may be entered in either upper or lower case; however, character strings and file parameters will not be translated.

Examples:

```
@LOWER
@L
```

3.2.7 UPPER

This command turns off a previous LOWER command. After executing this command, entered data will be translated to all upper case characters. EDT begins with UPPER set.

Example:

```
@UPPER
@UPP
```

3.2.8 DROP

Deletes all lines in every Proc file of EDT. This command resets the EDT pointers to the work space, so that the work space is effectively empty. This command actually releases unused disk space, whereas the DELETE command only marks lines as deleted.

Examples:

```
@DROP
@DR
```

3.2.9 SYSTEM

The SYSTEM command temporarily returns the user to monitor mode. None of the EDT work spaces are released.

If an operand is supplied, it will be treated as a SYSTEM command which will be executed immediately and control returned to EDT.

Examples:

```
@SYSTEM
@SY          STATUS TERM
```

The first example will interrupt EDT and allow entry of BEM commands. When the user wishes to return to EDT, the /RESUME command is used. The second example shows how a SYSTEM command may be entered.

NOTE: No slash is permitted on the SYSTEM command.

3.2.10 RSP

This is a special purpose command which terminates EDT and invokes the Remote Spool Processor (RSP). This command combines an @HALT command with a /EXEC RSP, with the exception that the EDT work space is passed to RSP. This means that the user may create data using EDT and then write that data to the OS/3 Spool file using RSP without writing the data to a library file.

Two conditions are imposed upon the use of this command.

1) RSP must be included in the BEM configuration. (See BEM — OS/3, *User Reference* UA-0139.)

2) The contents of all EDT proc files are passed to RSP along with the primary file. For example: if EDT's work space contained 50 lines and PROC4 contained an additional 15 lines, RSP would be passed to all 65 lines.

Example:

```
@RSP
```

3.2.11 FSTATUS

The EDT FSTATUS command is similiar to the BEM /FSTATUS command, except the file information is written to the EDT work space, rather than to the terminal. Ten-byte lines are written starting at the current line and incremented by the current increment. Each line contains the module type in column 1, a hyphen in column 2, and the module name in columns 3 through 8.

Format

@FSTATUS *library* [(*password*)] [,*volume*] [LONG]

where

library	Name of the file.
password	Read password for the file. It must be supplied with the command if the file was cataloged with a password.
volume	Name of the disk pack on which the file resides. If the file has been cataloged with a volume name, the parameter may be omitted.
LONG	Indicates that the comment information, date, and time of creation is to be appended to each record.

3.3 COMMAND ABBREVIATIONS

The following table illustrates the minimum abbreviations which may be used for commands. The user must enter at least those characters shown, but may enter additional characters to improve readability.

Command	Abbreviation	Command	Abbreviation
ALL	A	MOVE	M
BY	B	NUMBER	NU
COLUMN	COL	NOCHANGE	N
CONTINUE	CON	NOT	NOT
COPY	CO	ON	O
CHANGE	C	PAGE	P
CHAR	C	PRINT	P
DELETE	D	READ	R
DESEQ	DES	RSP	RS
DROP	DR	SEQUENCE	SEQ
FIND	FIN	SET	S
FIRST	F	STRIKE	ST
FSTATUS	FS	TABS	T
HALT	HA	TO	T
HELP	H	TYPE	TY
INSERT	I	UPDATE	U
LINE	L	UPPER	UPP
LOWER	L	WRITE	W

3.4 MULTIPLE COMMANDS

The following commands need not be the last command on a line:

```
HELP        ASSIGN      PROC
SET         DISPLAY     END
```

If a second command follows, it must be delimited by a space; no @ sign is needed after the first command.

Examples:

```
@SET P=23,L=70 PRINT 1-50
@ASSIGN G1='TEST' DISPLAY G1 DO 2
```

4 EDITOR PROCEDURE FILE COMMANDS

A Procedure file is a group of stored EDT commands which can be executed at any time by the user. This section describes the commands needed to create and execute a Procedure file. Although the commands are designed for use in Procedure files, they may be used elsewhere as well.

The PROC command tells the Editor which Procedure file is to be created. The END command indicates that the Proc file is completed and the Editor should return to normal processing mode. Each of the ten Proc files are identical; proc zero is the normal EDT Work file. All EDT commands are valid in any Proc file.

The DO command causes the specified Proc file to be read sequentially. Each line from the file is processed exactly as if it were typed from the terminal. Thus a sequence of EDT commands may be executed repetitively without transmitting each command separately each time it is needed.

4.1 DEFINING AND EXECUTING A PROCEDURE FILE

4.1.1 PROC [proc-number]

This command switches the user to the Procedure work file specified, or displays the current proc number. The proc number may be any single digit integer (0-9). Proc zero is the normal EDT work file.

```
@ PROC 1
```

would put the user in Proc file one. The first time a Proc file is entered, it will be empty; lines entered into a Proc file will remain until they are deleted or the Editor is terminated. Anytime a Proc file is entered, the current line number will be 1.0 even if the file is not empty. When in a Proc file, all commands refer only to that file and have no effect on any other Proc file.

```
@ PROC
```

displays the current Proc file number at the terminal.

The PROC command may not specify any proc that is currently an active proc as a result of a previous @PROC or @DO command.

4.1.2 END Command

This command returns the user to the previous proc file. The current line number is set to the line which was current in that proc file. That is, each time an @ PROC command is issued the current line and proc number are saved. An @END restores the last line and proc number that were saved.

An END command cannot be issued while in proc zero.

4.1.3 DO (proc number) [PRINT]

This command initiates the execution of a Procedure work file. Commands are retrieved from the file, optionally printed, and then executed. Upon completion of a command, the next command is retrieved from the file. This process is repeated until the last line is read from the file. Commands executed as a result of a DO command are processed exactly as they would be if typed from the terminal. Thus, if the user is currently in Proc file 2 and enters the DO command for Proc file 5, then commands retrieved from file 5 will act upon file 2.

```
@ DO 1
```

executes file 1.

```
@ DO 9 P
```

executes file 9 and prints each command line as it is processed.

If a DO command is issued from an executing proc file, control will be returned to the statement following the DO when the specified proc terminates. This allows ''subroutine'' type procs to be written and accessed from other procs.

4.1.4 RETURN Command

This command may be executed from a proc to terminate the execution of that proc. Its use is equivalent to executing the last statement in the proc file. If the proc issuing the RETURN command was initially called from another proc, then control will be returned to the calling proc; otherwise, control will revert to the terminal.

Example:

```
@RETURN
```

4.1.5 INPUT file parameters

This command is provided to permit user-written procs to be loaded and then executed with a single command. Command and text lines are accepted from the library file and processed as encountered. When processed, the entire file is read into a scratch space and then processed like a proc file. Consequently, any valid EDT command is permitted in the input file.

Example:

```
@INPUT COMMANDS,LIBFILE, PACK27
```

A typical input file might include commands and data statements as follows:

```
@PROC1
@@DISPLAY 'TEST OF INPUT'
@@PROC
@END
@PROC2
@@DISPLAY 'THIS IS PROC 2'
@@DO 1 P
@@DISPLAY 'END OF PROC 2'
@END
@DO 2
```

4.2 EDT VARIABLES

Ten EDT variables are available to hold values such as line numbers, column numbers, and search strings. Values may be assigned to variables before or during the execution of a Proc file to modify the operation of the proc.

A variable contains a character string of up to 50 characters; all variables are null until a value is assigned to them. When encountered in a command line the variable will be replaced by its current value before the command is parsed. Thus any part of a command may be specified by a variable symbol.

The variable is written as #Gn where n is an integer between 0 and 9. If the symbol # is to appear in a command line it must be written as ## so as not to be confused with a variable symbol.

EDT variables may be used in a variety of ways. They may be used to perform loops by assigning an initial value and incrementing it via ASSIGN statements and testing for loop completion via an IF statement. Within the loop the value of the variable may be referenced in commands using substitution as previously described. For example, if a series of commands were to be done on lines 1 to 20 of the workspace, the commands to do this would be similar to:

```
@ASSIGN G6 = 1
@(LOOP) IF G6 > 20       GOTO END
               .
               .
               .
          commands
               .
               .
               .
@PRINT #G6
@ASSIGN G6 = #G6+1        GOTO LOOP
@(END)     . . .
```

Variable replacement may also be used to change keywords or large portions of commands. For example, file parameters could be stored in a variable:

```
@ASSIGN G2 = 'ELT,MYFILE,MYPACK'
```

and each time they are needed in a command, only the EDT variable name need be typed:

```
@READ #G2
```

4.2.1 ASSIGN Statement

The Assign statement assigns a value to an EDT variable. There are several formats:

1. @ASSIGN Gn = 'String'
2. @ASSIGN Gn = n:i-j
3. @ASSIGN Gn = n[±m]
4. @ASSIGN Gn = Gm
5. @ASSIGN Gn = X'dddd'

Format 1. Any character string may be entered between the apostrophes. An apostrophe in the string must be entered as two consecutive apostrophes. A quote (") must be entered as two consecutive quotes. The length of the string is the number of characters between the apostrophes ('' count as one character). The maximum length of the string is 50 characters.

Format 2. Any character string is taken from a line in the EDT work space. n indicates the line number in the file; i and j represent the starting and ending column of the string. The length of the string is j-i+1.

Format 3. A numeric value is assigned to the variable symbol. n and m may be any valid line or column number. If either value contains a decimal point the result will also; otherwise the result will not. The different formats are needed to permit EDT variables to be used as column numbers which may not contain decimal points. A character value with all numerics is automatically converted to numeric format.

Format 4. The current value of the variable Gm is assigned to the variable Gn. In this format the value of the variable is assigned without any modification.

Format 5. Any hexadecimal string characters 0-9, A-F may be assigned.

NOTE: In Format 3 only numeric values may be assigned and the result is stored in a standard format; in Format 4 any value may be assigned.

The symbol Gn is used on the left-hand side of the equal sign in the ASSIGN statement and on the right-hand side in Format 4 only. All other variable references must be written with the symbol #Gn.

```
@ASSIGN G1 = 'STRING #G2'
@ASSIGN G3 = 12.5: 10-16
@ASSIGN G2 = #G9-#G8
@ASSIGN G6 = G7
@ASSIGN G9 = X '013AF0'
```

Unlike other EDT statements, the ASSIGN (and DISPLAY statements) need not be the last command or a line. Any EDT command may be entered following an ASSIGN or DISPLAY statement on a command line simply by leaving at least one space after the ASSIGN. (See also Section 3.4.)

```
@ASSIGN G1 = 10:1-3 DISPLAY '#G3' DO 5
@IF G1 = G2 DISPLAY 'TRUE' PRINT ?
```

The reason that the pound sign must not be used on the left side of the assignment is to avoid replacement when the ASSIGN statement is processed. If a pound sign were used, when EDT processed the command it would first perform substitution on all #Gn variables (including the one on the left side of the equal sign by substituting for the old value of the variable on the left side). Then EDT would try to execute the command resulting in unpredictable results. For example, if variable G4 contained the string G5, and the command

```
@ASSIGN #G4 = 'SNERD'
```

EDT would first perform substitution, obtaining:

```
@ASSIGN G5 = 'SNERD'
```

and then assign the characters 'SNERD' to variable G5.

4.2.2 DISPLAY Command

The DISPLAY command prints the value of an expression at the terminal. This command is useful when debugging procedures to display the value of a variable at a certain point or to display a message indicating the status of the proc. The expression displayed has the same format as the ASSIGN statement.

$$\text{@DISPLAY} \begin{Bmatrix} \text{'string'} \\ \text{n:i-j} \\ \text{n [}\pm\text{m]} \\ \text{Gn} \\ \text{X 'string'} \end{Bmatrix}$$

4.2.3 Variable Replacement

The first thing the Editor does upon recognizing a command line is to replace all EDT variables with their current values. When an EDT variable is found it is replaced with its string value; the command line is expanded to make room for the string. The number of characters required for a variable's value is given by its length. For a character value (Format 1 or 2 of the ASSIGN statement) the length is the string length; for a numeric value (Format 3) the length is either 9 or 4 depending on whether or not a fractional value is specified.

A portion of a string may be indicated by specifying a starting position and a length:

```
#G3(s,l)
```

where

s	Indicates the starting position $1 \leqslant s \leqslant 50$
l	Indicates the length $1 < s + l \leqslant 51$

The values of s and l may be specified in terms of #Gn symbols.

If '#G3' is 'ALPHABET' with length = 8
then '#G3(2,5)' will be 'LPHAB' with length = 5.

If the specified length is greater than the implied length of the variable, blanks are appended to the right side as needed.

Thus '#G3(6,8)' = 'BET

For example,

```
@ASSIGN G1 = 1
@A   G2=.0005
@A   G4='SEARCH'
@A   G3=#G1 + #G2
```

```
then @ON   #G1 — #G3 COL #G1 PRINT '#G4'
becomes @ON 0001-0001.0005   COL 0001 PRINT 'SEARCH'
```

4.2.4 EDT Variable Usage

In any command line, an occurrence of the symbol #Gn is replaced with its current value. Expressions in ASSIGN, DISPLAY, and IF statements are evaluated after variable values are substituted, thus some expressions can result in command line overflow on these commands. In these commands, any string value may appear between quotes, but only numeric values may appear without quotes. If an operand consists of simply Gn, then the current string value of #Gn is used.

For example, suppose @A G1='10'
@A G2='10.0'

1) then @IF'#G1'='#G2' DISPLAY 'TRUE'

2) and @IF G1=G2 DISPLAY 'TRUE'

are equivalent and FALSE

3) but @IF#G1=#G2 DISPLAY'TRUE'

is TRUE because numeric values are used rather than character strings.

1) String values are replaced before the comparison so the statement is actually @IF '10'='10.0' DISPLAY 'TRUE'. These two character strings are not equal.

2) The current string values are compared.

3) Substitution is performed.

@IF 10 = 10.0 DISPLAY 'TRUE'

Quotes are not specified so the operands are treated as numeric values and the comparison is true.

The two formats:

1) @DISPLAY'#G1'
2) @DISPLAY G1

are equivalent. However, the first is a more general format allowing strings, substrings, and variables to be displayed. The second format will only display an entire variable symbol.

For example, the first format can include character constants and other values:

@DISPLAY 'VALUE 1 = #G1 AT LINE #G2'

Because all numeric values are considered to be line or column numbers, zero or negative values are not permitted in numeric expressions.

4.3 CONTROLLING SEQUENCE OF EXECUTION COMMANDS

These commands allow branching within an EDT Procedure file.

4.3.1 GOTO

This command causes an unconditional branch to another statement in the procedure file. The statement may be specified either by its absolute line number or by a symbolic label.

```
@GOTO  10
@GO    213.1
@GOTO  END
@GOTO  A
```

Symbolic labels must consist of one to four alphanumeric characters, the first of which must be alphabetic.

Labels may be defined on any EDT command in the procedure file. Labels are enclosed in parentheses and must start in column two of the command line.

```
@(A)     PRINT 10-20
@(END)   NOP
```

4.3.2 IF

This command allows an EDT command to be executed conditionally based on some condition. To test the success of the last search command (any INSERT, CHANGE, SEQUENCE, or any use of search-string) the operand is .T. for a successful search, .F. for an unsuccessful search. To test other conditions the format is <expr> <op> <expr> where <expr> is the same as the operand of an ASSIGN or DISPLAY command and <op> is any combination of the relational operators <(less than), >(greater than), and = (equals).

Any command may be executed in an IF statement, including another IF statement.

For example:

```
@IF'#G1' = '#G2'GOTO 1
```

transfers control to line 1 if the two strings are identical.

```
@IF#G1<>#G2     PRINT'#G3'
```

prints occurrences of the string given by #G3 if the numeric values for #G1 and #G2 are not equal.

```
@IF'#G7'>'Z'   GOTO 2
```

branches whenever #G7 is greater than Z. Comparison assumes that blanks are not significant. Thus, if in the preceding example #G7 = 'A ' then the relation will not be satisfied (i.e., A is not greater than Z).

4.3.3 NOP

This command provides a NO-oPeration command for defining extra line numbers for branching. It may also be used to enter comments in a Procedure file.

```
@NOP ANY COMMENTS MAY APPEAR HERE
```

4.4 COMMAND ABBREVIATIONS

The following abbreviations are used for the Editor Procedure file commands:

Command	Abbreviation
ASSIGN	A
DISPLAY	DI
DO	DO
END	E
GOTO	G
IF	IF
NOP	N
PROC	PRO
RETURN	RET

4.5 PROCEDURE EXAMPLES

This section gives a number of examples of EDT procs with an explanation of each line in the example. These examples are intended both as a learning device and to make commonly used procs available to the general user.

4.5.1 RPG II Proc

This proc assists in the entry of RPG II programs. It displays column numbers across the top two lines of a UNISCOPE screen. (The example assumes a U200 with dimensions of 24×80, but it is easily modified for other scopes.) As lines are typed they are stored at the current location in the work file and the screen is rolled down. The UNISCOPE display thus shows the most recently typed line with the highest line number on the third line of the screen.

```
(01)      @PROC 1
(02)      @DELETE
(03)      @SET LINE=128
(04)      @SYSTEM SCREEN NOROLL
(05)      @@ASSIGN G1=*
(06)      @@0.0001
(07)      XX         1         2         3         4         5         6         7         8
(08)      XX.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.YY
(09)      @@CHANGE FIRST 'XX' TO X'100301010000'
(10)      @@CHANGE FIRST 'XX' TO X'100102010000'
(11)      @@CHANGE FIRST 'YY' T X'1E'
(12)      @@P .0001-.0002
(13)      XX
(14)      @@CHANGE FIRST 'XX' TO X'100103010000'
(15)      @@(LOOP)DISPLAY X'10010301000027910000100117010000'
(16)      @@COPY  .0003 TO #G1
(17)      @@UPDATE #G1
(18)      @@IF #G1:1-2 = '/*' GOTO END
(19)      @@ASSIGN G1=#G1+1 GOTO LOOP
(20)      @@(END) DELETE #G1,.0001-.0003
(21)      @@#G1
(22)      @END
(23)      @DO 1
```

The preceding proc is written in the form of an INPUT file. If it appears in a library file exactly as shown, it may be loaded by issuing an @INPUT against it. Lines which contain a single at sign (@) will be executed as the file is read by INPUT; those with two at signs (@@) will be treated as data lines by INPUT and will be executed when an @DO is issued against the proc. A line by line description of the proc follows.

Line	Description
1	Sets EDT to proc number 1. Following commands and data will operate on that proc. This command is executed as soon as INPUT reads it.
2	Deletes the contents of proc 1.
3	Sets the EDT line width to its maximum value of 128.
4	Turns off the BEM screen roll mode, so that the proc may do its own scrolling.
5	The first line that is actually stored in the proc work space. It will be executed when an @ DO1 command is issued (line #23 of INPUT file). When executed this command will initialize the variable G1 to the current line pointer.
6	Sets the current line pointer to an unused line at the start of the file. This line will be used as scratch space by the proc.
7	This line is a data line. It will be used as the top header line on the screen.
8	This line is also a data line. It will be the second header line.
9	Edits the data line just entered. It changes the string XX at the start of the line (#7) to a DICE code sequence. These codes position the cursor at the top of the screen and erase the screen. The data line is thus displayed on the top line of the screen.
10	Edits the second data line (#8). It inserts the DICE codes to position the cursor at line 2 of the screen.

Line	Description
11	Adds a start of entry (SOE) character at the end of the second header line.
12	Prints the two header lines which were formatted by the previous commands (#7 through #11). Note that the EDT line numbers which normally precede each line displayed will not be seen because the UNISCOPE cursor is repositioned after the line is displayed.
13	Creates a dummy line at 0.0003 of the work space.
14	Changes the line just created to the DICE codes to position the cursor at line 3 of the UNISCOPE screen. Change commands are used throughout this proc to create DICE codes rather than having the DICE codes contained in the data lines so that the PROC will be readable and can be displayed without having the DICE codes processed.
15	This is the first line of the proc's loop. It displays a hex string which positions the cursor at line 3, scrolls the screen down one line, and then moves the cursor to line 23 of the scope.
16	Creates a copy of the line formatted by line 14 at the next location in the user's work space.
17	Causes the line copied on line 16 to be displayed for update. Its current contents just position the cursor to line 3. The terminal user can then type in the contents of the line being created and transmit it. The transmitted line will replace the line containing DICE codes.
18	Tests the contents of the line received from the UPDATE. If it contains the stop symbol, the GOTO command is executed and control goes to the statement in the proc file which contains the statement label END . In this example the stop symbol is /* in columns one and two, however any symbols which can never occur in the data lines may be used.
19	This statement increments the line pointer (G1) to the next line to create. Notice that the symbol # is only used on the righthand side of the equal sign. Also on this line is a GOTO statement which transfers control to the statement with the label LOOP (line 15).
20	Control passes to this statement when the stop symbol is encountered in line 18. This command deletes several lines which should not be in the user's work space. These are the last line updated, which contains the stop symbol, and the three data lines at the start of the file which were used for temporary storage by the proc while it executed.
21	Sets the current line pointer to the line which would have been entered next had the stop symbol not been entered. This allows the user to restart the proc where he previously left off simply by entering the command @ D01.
22	This is the first statement since line 4 which is executed during the @INPUT command. The intervening statements were all processed as data by the @INPUT processor. This statement terminates proc 1 and returns to the primary work space which will be the user's data area.
23	Initiates the proc which was just loaded.

After the INPUT command is executed, proc 1 will contain the following lines:

```
0001.0000       @ASSIGN G1=*
0002.0000       @0.0001
0003.0000       XX        1         2         3         4         5         6         7         8
0004.0000       XX.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.0.2.4.6.8.YY
0005.0000       @CHANGE FIRST 'XX' TO X'100301010000'
0006.0000       @CHANGE FIRST 'XX' TO X'100102010000'
0007.0000       @CHANGE FIRST 'YY' T X'1E'
0008.0000       @P .0001-.0002
0009.0000       XX
0010.0000       @CHANGE FIRST 'XX' TO X'100103010000'
0011.0000       @(LOOP)DISPLAY X'10010301000027910000100117010000'
0012.0000       @COPY  .0003 TO #G1
0013.0000       @UPDATE #G1
0014.0000       @IF #G1:1-2 = '/*' GOTO END
0015.0000       @ASSIGN G1=#G1+1 GOTO LOOP
0016.0000       @(END) DELETE #G1,.0001-.0003
0017.0000       @#G1
```

This proc is intended to simplify the entering of RPG II programs by always displaying column numbers for use as guidelines in the typing of fixed field cards. The proc itself is insensitive to the data entered and may be modified by the user to suit his own needs.

4.5.2 Selective FSTATUS Proc

This proc will do an FSTATUS command for a specified file name. It will then delete all load and object module names from the list and sort the source module names to the front of the file. It then will request a search-string and will print all element names which contain that string.

```
(01)     @DELETE
(02)     @0.1:ENTER FILE NAME >
(03)     @ON 0.1 C '>' T X'1E'  UPDATE
(04)     @ASSIGN G1=0.1:1-50
(05)     @FSTAT #G1
(06)     @ASSIGN  G3=*
(07)     @ASSIGN  G2=#G3+1
(08)     @COL1-2 DELETE 'L-'
(09)     @COL1-2 DELETE 'O-'
(10)     @COL 1-2 ON 1-#G3 MOVE 'P-' TO #G2
(11)     @0.1:ENTER SEARCH STRING >
(12)     @ON 0.1 C'>' TO X'1E' UPDATE
(13)     @ON 0.1 FIND ' '
(14)     @ASSIGN G0=%-1
(15)     @ASSIGN G1=0.1:1-#G0   DELETE 0.1
(16)     @PRINT '#G1'  COL 3-10
(17)     @RETURN
```

Line	Description
1	Deletes the contents of the work file.
2	Creates a string in line 0.1 which will be used to ask the terminal user a question.
3	Creates an SOE on the created line and displays the line to the user. The user's reply is accepted as the updated line 0.1.
4	Sets the variable G1 to the user's reply which is the name of the file whose directory is to be listed. It also deletes the reply line.

Line	Description
5	Does the actual FSTAT of the file. The element names are listed, one to a line, in the EDT work file.
6	Sets G3 to the current line number, which is one greater than the number of elements in the file.
7	Sets G2 to a value greater than G3 so that an overlapping line range is never specified in line 10.
8	Deletes all load module entries from the work space.
9	Deletes all object module entries from the work space.
10	Moves all proc or macro entries (identified by the type in column 1) to the end of the file.
11	Creates a line containing a question for the terminal user.
12	Adds the SOE character and solicits the user's response.
13	Finds the end of the search-string specified by the user. The variable % is set to the first space on the line.
14	Sets G0 to the last nonspace on the line.
15	Sets G1 to the search-string entered by the user and then deletes the reply line from the file.
16	Prints all the elements which contain the search-string as a part of the element name.
17	Terminates the proc. It is optional here because executing the last line in a proc always terminates the proc.

5 ERROR DETECTION AND RECOVERY

5.1 ERROR MESSAGES

A complete list of error messages is provided in Appendix C.

When an error message is printed, the Editor displays the current line number on the next line. Generally an error which occurs during command execution terminates the command.

All error messages are displayed through BEM's message display facility.

Messages are designed to describe the situation causing the error. For a description of what caused the error, use the @HELP command after the error.

There are several classes of errors which may be encountered. Command errors occur when an invalid command is entered by the user. These may be detected as soon as the command is entered or after the command has begun to be processed, depending on the nature of the error. Unrecognizable commands or invalid command combinations are detected immediately. Errors in line and column ranges will not be detected until the command has processed up to that range. Consequently, a command may be partially completed before an error is detected.

Other errors result from file errors. When an error occurs during a WRITE command, the file directory is not updated, thus no part of the element is stored in the file.

Errors resulting from the work file are usually unrecoverable. The user should attempt a retry, and try to save his work area. OS/3 operator intervention may be required.

6 BEM OPERATION

6.1 INTRODUCTION

This section instructs the novice user how to use BEM and how to execute EDT in particular. The purpose of this section is to explain commands available to the terminal user, how to initiate a session, execute and monitor application programs, and terminate the session. For details on how to configure BEM, console operation, etc., the user should consult the *BEM — OS/3 Basic Editor Monitor, User Reference* UA-0139.

To use BEM, the user must locate a free terminal and log on. Logging on consists of entering the LOGON command, together with a user-id, account number, and password. This user-id is used for identification by the console operator, the account number is used for billing purposes, and the password for security; each is one to four characters long, and either of the last two may be omited depending on conditions at your site. When the LOGON command is accepted, BEM will display the log-on bulletin and inform the user it is ready to process requests. The log-on bulletin is built by the administrator and may contain messages to inform the users of resource availability or system status.

The administrator may also assign a default file to each user's account, and place certain restrictions on command and file usage. If the user wishes to use the default file, the file and volume names should be omited from the command which references the file. If a user is informed that he cannot access a certain file, or is not permited to write to a certain file, the system administrator will need to be contacted to remove the restrictions.

Once logged on, the user is placed in monitor mode and may enter any monitor command. To identify monitor mode, BEM presents the UNISCOPE start of entry (SOE) character, followed by a slash (/). All monitor commands begin with the slash, but if the user does not erase the screen, the slash is supplied by BEM.

Several commands may be entered while in command mode. The HELP command functions identically to the EDT HELP command. TYPE allows the user to send a message or question to the 90/30 console. Three STATUS commands are provided, one to list users of BEM, another to list OS/3 resources available and those in use, and the last gives information about the user's own terminal.

The EXECUTE command is used to load and run BEM application programs such as BASIC and EDT. The program is located and loaded, and any additional storage (memory or disk) requests for that program are processed. Once loaded, the program is in control of the user's terminal and all key-ins and responses are controlled by it.

While a program is processing, BEM provides a method to interrupt it and return control to the monitor. To interrupt a program, the user hits the MESSAGE-WAIT-ING key or transmits anything. A program may only be interrupted when it is active. If it is awaiting input, the program provides its own way to exit to the monitor; for example, the SYSTEM command may be used to interrupt EDT.

One advantage of this capability is that during execution a proc may be interrupted to enter a monitor command, then the proc may be resumed using the RESUME command. The RESUME command will not function if the EXECUTE or LOGOFF commands are entered since the user's work areas are destroyed.

At the end of a session the LOGOFF command is used to release all storage that has been acquired during the session. To begin a session after LOGOFF has been processed, the next user must use the LOGON command again.

6.2 COMMAND FORMAT

All BEM commands begin with a slash (/), and are immediately followed by the command keyword. The slash is normally provided by BEM, but it must be entered if the user has altered the screen, or is operating a nonvideo terminal. Commands may be abbreviated by typing at least those characters which are underlined.

6.2.1 LOGON

The LOGON command is used to begin a BEM session. The "id" used may be one to four characters, and is determined at the user site.

Format

/LOGON *user-id,[account-number][,password]*

6.2.2 HELP

The HELP command allows the user to obtain additional information or explanation about an error or status message which has just been displayed. The HELP command should be entered immediately after the message requiring explanation, since the command always relates to the message immediately preceding the HELP query.

Format

/HELP

6.2.3 TYPE

The TYPE command is used to send a message or question to the console operator. All characters following the command keyword are sent to the 90/30 console. Up to 52 characters may be sent.

Format

/TYPE *comment*

6.2.4 PAUSE

The PAUSE command is used to send a question to the console operator. It is different from the TYPE command in that the user's task is suspended until the operator replies to the message.

Format

```
/PAUSE     question
```

6.2.5 STATUS

Three formats of this command are available. The first will display information concerning terminals on the system. The second format will display information about 90/30 resources in use by BEM. Lastly, information about the user's own terminal may be obtained.

Format 1

```
/STATUS    TERM
```

Output Format 1:

```
TERMINAL    COMMAND    PROGRAM    SCRATCH SPACE    ID

  nnnn       cccccc     pppppp         sss         uuuu
    .          .          .             .            .
    .          .          .             .            .
    .          .          .             .            .
```

where

nnnn	Terminal name in the form T*lt*0 where *l* is the line number and *t* is the terminal number.
cccccc	Last system command issued at this terminal.
pppppp	Last program executed at this terminal.
sss	Number of disk scratch space cylinders acquired by this terminal.
uuuu	User's identification code.

Format 2

```
/STATUS RESOURCE
```

Output Format 2:

```
                    ------------MEMORY------------    -----SCRATCH-----

TASKS    TERMS    MAX       AVAIL    FREE             MAX      FREE
 nnn      ttt     mmmmm     aaaaa    fffff            sss      ddd
```

where

nnn	Total number of tasks which may be active at one time. This is the maximum number of terminals which may be logged on the system.
ttt	Number of terminals currently logged on
mmmmm	Amount of storage, in bytes, allocated to the entire BEM system as obtained from the job card
aaaaa	Total amount of storage, in bytes, available for allocation to users and program areas
fffff	Current amount of storage, in bytes, which is free to be allocated
sss	Total number of disk cylinders available for allocation to users and programs
ddd	Current number of disk cylinders which is free to be allocated

The third STATUS option will display the user's id, terminal number, logon time, current date, and wall-clock time.

Format 3

/STATUS

Output Format 3:

```
TERMINAL    USER    LOGON    DATE        CUR-TIME
  E001      PROC    09:07    78/02/17    09:08:20
```

6.2.6 EXECUTE

This command is used to invoke application programs. Sufficient memory and disk space must be available, if required by the program, for loading to complete successfully.

Format

/EXECUTE *program*

Programming Note

Programs which may be executed are EDT, RSP, and BASIC.

6.2.7 LOGOFF

The LOGOFF command is used to terminate a session. All work areas assigned to the user are released.

Format

```
/LOGOFF
```

6.2.8 FSTATUS

To obtain a directory listing of an OS/3 library file at the terminal, the file status (FSTATUS) command may be used. This command will display the name of each source, proc, object, or load module, and the type of each module. An alternate form of this command (LONG), displays the additional information about each module.

Format

/FSTATUS   *library*[(*password*)] [,*volume*] [LONG]

where

library	Name of the file which is to be listed.
password	Read password for the file. It must be supplied with the command if the file was cataloged with a password.
volume	Name of the disk pack on which the file resides. If the file has been cataloged with a volume name, the parameter may be omitted.

This command will produce output similar to:

```
P-SUPEQU    P-EOJ      S-SRCMOD    S-COPYMOD
S-COBOLPRG  P-CLOSE    O-OBJMOD    L-LODMOD00
```

To obtain a directory listing of the default file for an account (if one exists), enter the command without any operands.

The LONG format of the FSTATUS command produces output similar to:

```
P-SUPEQU      SUPERVISOR EQUATES       02/05/78    12:15
P-EOJ         END OF JOB PROC          01/31/77    02:59
S-SRCMOD      COBOL PROGRAM            07/14/77    14:20
S-COPYMOD     COBOL COPY MODULE        07/14/77    14:35
S-COBOLPRG                             07/14/77    15:05
P-CLOSE       CLOSE THE FILE           01/28/77    22:06
O-LOAD        PROGRAM TO SAVE FILE     09/15/78    08:15
L-LOADMOD     PROGRAM TO SAVE FILE     09/15/78    08:17
```

If the LONG format is used with the default file, a single comma must precede LONG:

```
/FSTAT , LONG
```

6.2.9 PRINT and PUNCH

These two commands may be used to produce a printed listing of a module, or a punched card deck. The PRINT command will list a module on the system printer. A heading identifying the user and line numbers are also produced. The PUNCH command will punch the named module on the system punch. Identifier cards are punched preceding and following each deck to give the user's task information.

Format

/PRINT *element,file*[(*password*)],[*volume*] [,*type*]
/PUNCH *element,file* [(*password*)], [*volume*] [,*type*]

where

element	Name of the module to be printed or punched.
file	Name of the OS/3 library file which contains the element.
password	Read password for the file. It must be included in the command if the file has been cataloged with a password.
volume	Name of the disk pack on which the file resides. If the file has been cataloged with a volume name, this parameter may be omitted.
type	Element-type of the module. An S denotes source, a P denotes proc. If this is omitted, source is assumed.

6.2.10 DELETE

This command may be used to delete an element from a library file. Any macro proc, source, object, or load element or group header may be deleted.

Format

/DELETE *element,file*[(*password*)], [*volume*] [,*type*]

where

element	Name of the module to be deleted.
file	Name of the OS/3 library file which contains the element.
password	Write password for the file. It must be included in the command if the file has been cataloged with a password.
volume	Name of the disk pack on which the file resides. If the file has been cataloged with a volume name, this parameter may be omitted.
type	Element type of the module:

S	Source
P	Proc
M	Macro
O	Object
L	Load
G	Group header

If this is omitted, source is assumed.

NOTE: Type G specifies that only group headers (BOG and EOG markers) be deleted, not the entire group.

6.2.11 RUN

This command will schedule a batch OS/3 job. If a job name is specified the job control is assumed to be stored in the system Job Control file ($Y$JCS). If the name is omitted, the job control is assumed to be in the JCS queue of the system Spool file.

Format

/RUN *program*

Example:

```
/RUN LISTLIB
/RUN
```

When a job is scheduled via BEM, the user will not be notified of its actual initiation or termination. The DISPLAY JOBS command may be used at the terminal to monitor the execution of a batch job.

NOTE: The RUN command is an optional feature of BEM and may not be available at your site due to operating procedures.

6.2.12 DISPLAY

The DISPLAY command gives information about OS/3 System usage. This command takes two forms:

- Information about batch jobs
- List of disk volumes currently mounted

Information about batch jobs may be obtained using the JOBS DISPLAY option.

Format

```
/DISPLAY JOBS
```

Examples:

```
Job Name        Size      Time      Step      Exec      Job No.

BEM             044956    13.2      01        BEM000    0002
ASMTEST         131072    25.8      02        ASM000    0015
FREE MEMORY     004096
```

where

Job name	The name of each batch job currently executing.
Size	Amount of memory allocated to that job including program load area and job prologue in decimal.
Time	Current elapsed CPU time for all steps of job in seconds.
Step	Step number currently executing.
Exec	Name of current load module.
Job no.	Unique job number assigned by spooling.

The unused memory entry shows total free memory at the time of the DISPLAY. Memory allocated to the supervisor, symbionts, and ICAM is not explicitly shown by the display.

A list of disk volumes on the OS/3 System may be obtained with the VOLUMES option.

Format

$\text{/}\underline{\text{DI}}\text{SPLAY }\underline{\text{V}}\text{OLUMES}$

Example:

```
*OS3REL   USER01   *BEMPAK
```

This example shows three disk packs mounted. The two that are accessible to the BEM user are marked with an asterisk (*).

6.2.13 SCREEN

The SCREEN command is used to inform the BEM System of certain UNISCOPE characteristics or options which the user wishes to utilize.

Format

$$\text{/}\underline{\text{SC}}\text{REEN } [\textit{dimension}] \left[, \begin{array}{l} \underline{\text{C}}\text{OP} \\ \underline{\text{NOC}}\text{OP} \end{array} \right] \left[, \begin{array}{l} \underline{\text{R}}\text{OLL} \\ \underline{\text{NOR}}\text{OLL} \end{array} \right] [\underline{,\text{U}}\text{TS400}]$$

where

dimension	Size of the UNISCOPE screen: height × width (e.g., 16×64, 24×80). For a hardcopy device, the width is ignored, but the height will control the number of lines printed at a time.
COP	Indicates that all messages output by BEM are to be logged on the COP printer.
NOCOP	Messages are no longer to be logged.
ROLL	All messages displayed by BEM will be displayed at the bottom of the UNISCOPE screen; the screen will be scrolled up.
NOROLL	UNISCOPE screen will no longer be scrolled.
UTS400	Indicates to BEM that this is a UTS400 terminal and sets the UTS control page for correct RSP operation.

The COP option should not be used unless the device is actually present and configured, or control will not be returned to the terminal. If such a problem occurs, the user should clear the terminal, and issue a /SCREEN NOCOP command to restore operation.

The COP option provides the ability to obtain selected hardcopy listings at the terminal. It is not intended to produce a hardcopy log of all terminal transactions. Consequently, not all BEM commands will produce meaningful COP listings. To get a hard copy of an FSTATUS or DISPLAY, for instance, the user should format the screen and use the UNISCOPE terminal PRINT button.

The ROLL option truncates all output to a single line on the UNISCOPE screen and thus should not be used when longer lines need to be displayed. Two lines are always left at the bottom of the screen, however, for data entry.

6.2.14 VTOC Command

The VTOC command may be used to display the names of the files on a disk volume. The name of each file on the disk will be shown, along with the number of cylinders allocated to the file, the file type, and extent count. If the file is a library file (file type = SAT), additional information is displayed showing the remaining free space in each partition of the file. This command may be issued to any disk allocated to the BEM system.

Format:

/<u>V</u>TOC *volume-name*

Example:

```
/VTOC PACK22
```

would produce output similar to:

```
FILENAME          CYL.   EXTENTS      TYPE     DIRECTORY/     DATA/   B-LOAD
SAMFILE           010      01         SAM
RANDFILE          002      01         D.A.
LIBFILE           050      05         SAT          124/        4021/      4
VERYLONGFILENAME
                  010      02         SAT            0/           0/      0
```

6.2.15 Disk Space Management Commands

These commands allow the user to create and erase files dynamically under BEM. As with most other BEM commands, their use may be restricted by the system administrator for certain accounts.

6.2.15.1 ALLOCATE COMMAND

This command will allocate a new disk file on a specified volume. The file may be any OS/3 file type. If it is a SAT file, it may be initialized as an OS/3 library file.

Format

$$/\underline{A}LLOCATE\ \ \textit{type, file-parameters}\ \left[,\underline{INIT} = \begin{Bmatrix} YES \\ NO \end{Bmatrix}\right] [,\underline{S}IZE = n]\ [,\underline{INC} = n]$$

where

type	Indicates the type of file to be allocated: ST — SAT (possibly a library file) IR — IRAM IS — ISAM DA — Direct access SQ — Sequential NI — Nonindexed
file-parameters	Valid OS/3 file description of a file which does not exist on the volume. The volume stated in the parameter list specifies where the file will be placed.
INIT	YES — Causes the SAT file to be initialized as an OS/3 library file. This is the default value for a SAT file. NO — The file is not initialized. This is the default value for non-SAT files.
SIZE	Initial allocation SIZE in cylinders. Default value is ten cylinders.
INC	SIZE in cylinders of any extents added when the file is extended. Default is one cylinder.

Any DA, SQ, or NI files allocated may be processed by BASIC. Any initialized SAT file may be processed as a library file by any BEM module.

6.2.15.2 SCRATCH COMMAND

This command will scratch any file except system files. If the file is catalogued, its catalog entry will not be removed. The user should be careful when using this command, as once a file has been scratched, its contents are inaccessible.

Format

$/\underline{SC}RATCH$ *file-parameters*

where

file-parameters	Description of the file to be scratched. This may not be a Y file.

6.2.16 ENTER Command

This command enters an OS/3 library file element to be executed in BEM background mode. This function is only available if it is configured by the system administrator. Tasks entered in background are executed by BEM exactly as from interactive terminals except that output is produced on the high speed printer.

Format

/ENTER *element,file-parameters* [,*type*]

where

element	Name of the module to be entered.
file-parameters	Description of the OS/3 library file which contains the element.
type	Element-type for the module. An S denotes source, a P denotes proc. If this is omitted, source is assumed.

The ENTER facility allows users to submit an OS/3 library element containing commands and data just as they would be entered at the terminal. This element may contain one or more LOGON-LOGOFF sequences, and each task (LOGON-LOGOFF pair) may perform any functions which would be valid at the terminal. The first statement of an entered deck must be a LOGON command, and there should not be any cards between the LOGOFF and LOGON commands when several tasks are stacked in a single ENTER deck.

Decks submitted via the ENTER function are queued in the OS/3 spool file, along with background decks submitted through the card reader. These decks are then processed in a first-come first-served manner concurrently with interactive processing. The number of tasks available to process these decks is defined by the system administrator; more than one background task may be active at a time.

Output from entered tasks is routed to the main site printers and each task's output is identified with the user-id from the LOGON statement. Invalid LOGON statements in a deck cause BEM to begin rejecting cards until a valid LOGON is found, or the end of the deck is reached. Rejected cards are printed on a separate listing.

Each time an input is expected during a background session, BEM attempts to read the next card. This card could be either a command or a line of data. It is processed just as if it had been entered from a terminal. If an error is encountered during the processing of a command, the error message is printed and processing continues with the next card; the session is not aborted. The only condition which will cause a background session to be aborted is the exhaustion of all input. This is usually due to a missing or misinterpreted LOGOFF statement, and results in the task being logged off.

Certain conditions which normally arise at a terminal have been modified for background tasks. These are:

- CONTINUE queries are eliminated for background tasks and all output is displayed in its entirety. Normally, BEM outputs one screen of lines and suspends output until the user answers the CONTINUE query.
- OVERWRITE queries are eliminated for background tasks. If a module to be written already exists, it is deleted and a new one written automatically.
- OUT OF MEMORY conditions for background tasks are considered errors and a NO response is assumed.
- Batch tasks are treated as hardcopy terminals, thus RSP is not available.

6.2.17 COMMENT Command

This command permits the user to enter comments in the comment field associated with an OS/3 library element. The element is located, and then the 30-character comment specified in the command is applied.

Format

/<u>C</u>OMMENT *element,file-parameters* [,*type*] *comment*

where

element	Name of the OS/3 librarian format element to be commented.
file-parameters	Specifies the location of the file containing the element.
type	Specifies the element type. A P denotes proc or macro; an S or blank, source; an O, object; an L, load.
comment	A 30-character string to be used as a comment. It must be separated from the file-parameters by exactly one space. Any additional spaces are considered part of the comment.

6.2.18 BULLETIN Command

This special purpose command allows the system administrator to read, display, and change (using the WRITE keyword) the LOGON bulletin. The READ and WRITE options are restricted to privileged users only, while DISPLAY can be used by any user.

Format

```
/BULLETIN     ⎧ READ    ⎫
              ⎨ DISPLAY ⎬
              ⎩ WRITE   ⎭
```

where

/BULLETIN READ	Deletes the entire contents of the user's workspace and then reads the current LOGON bulletin into the workspace. This command should be issued while in EDT or RSP as a SYSTEM command to avoid losing the workspace again on entry into EDT or RSP. *NOTE: This option is equivalent to @DROP; all procs are lost.*
/BULLETIN DISPLAY	Displays the current LOGON bulletin to the terminal. This option can be invoked by any user.
/BULLETIN WRITE	Overwrites the existing LOGON bulletin with the contents of the user's workspace. If the entire new bulletin will not fit in the maximum space reserved for LOGON bulletins, only as much as will fit is written and an error will be displayed. The user can find out how much was accepted via the BULLETIN DISPLAY function. It is allowable to write a new bulletin which is larger than the existing one, provided the maximum bulletin space limit is not exceeded. This command should be issued from EDT or RSP via a SYSTEM command.

6.2.19 RECOVER Command

This command allows the terminal user to recover OS/3 librarian elements which were unintentionally deleted. It is only effective for elements which have been deleted recently and have not been entirely removed from the file via a PAC librarian statement. It must be used carefully to ensure that the correct element is "undeleted" (there may be several to choose from).

Format

/RECOVER *element,file-parameter*[*,type*]

where

element	Name of the deleted modules to be recovered.

file-parameters	Location of the file containing the elements to be recovered.
type	Element type which is to be used to rebuild directory entries for the deleted element.

Once invoked, this command will begin by listing each deleted element which could possibly have the same element type specified in the command. For example, if the user attempts to recover a source module named TEST, and the file contains both source and load deleted modules, only the source modules will be shown:

```
      /RECOVER     TEST,MYFILE,MYPACK,S

  1.    TEST     OS/3     TEST PROGRAM     01/30/78     12:48
  2.    TEST     OS/3     TEST PROGRAM     01/30/78     14:02
  3.  * TEST     OS/3     TEST PROGRAM     01/30/78     15:25

  SELECT NUMBER AND NEW NAME ▷
```

Each element with the name and type indicated will be displayed, with a sequence number for identification purposes. The comment field, date, and time of creation will also be shown. If an undeleted element currently exists, it too will be shown and flagged with an asterisk.

After displaying the list, the user will be asked to select an element (by number), and the name under which the recovered element is to be written. The name selected by the user must not be a name which already exists. As long as this rule is followed, the user may rename any of the modules listed including the active one; thus RECOVER may be used to rename modules too.

Continuing with this example, if the user wished to retain the active TEST element, but also recover deleted element 2, the response

```
      ▷2,TEST2
```

could be entered to recover copy 2 of TEST and rename it to TEST2. BEM will insure that another module of the same name does not already exist and generate appropriate error messages.

If, on the other hand, the user did not want the active copy of TEST (#3), but wished to restore copy two, he could rename the active copy and restore copy two via:

```
      ▷3,DUMMY
      ▷2,TEST
```

and later go back and delete element DUMMY from the file.

Each time the user renames a module, BEM will list the elements again with new numbers to avoid confusion. To end the RECOVER command, type STOP.

6.3 BATCH SUBMISSION

An optional feature available at some sites is the capability for entering card decks of BEM sessions for background execution. This feature permits access to the system when a terminal is not available.

To use the batch capability, the user need only keypunch the session from LOGON to LOGOFF, and submit it to BEM via the computer operator. The deck will be queued and executed on a first-come first-served basis.

Output from batch tasks is routed to the main site printers, and each task's output is identified with the user-id from the LOGON statement.

Batch decks are processed in a manner similar to the decks submitted via the ENTER facility. For additional details on how these decks are processed, and how errors are handled, see the description of the ENTER command (6.2.16).

APPENDIX A EDT COMMANDS

Table A-1 EDT Commands

Command	Function	Functional Category	Section
ALL	Modifies the CHANGE and INSERT commands to operate on the entire column-range specified rather than terminating with the first replacement.	Line and column range specifications	3.1.1.1
ASSIGN	Assigns value to EDT variable.	Procedure file	4.2.1
BY (increment)	Specifies the increment to be used for the SEQUENCE command.	Line and column range specifications	3.1.1.2
CHANGE	Replaces first occurrence of search-string with change-string in the given column range of each line in the line range.	Data manipulation operations	3.1.2.1
COLUMN (column-range)	Defines column range.	Line and column range specifications	3.1.1.3
COPY	Duplicates a specified range of lines or propagates a line or range of lines.	Data manipulation operations	3.1.2.2
DELETE	Deletes selected lines from the EDT work area.	Data manipulation operations	3.1.2.3
DESEQ	Take line numbers from data lines in READ.	Line and column range specifications.	3.1.1.11
DISPLAY	Assigns value to EDT variable.	Procedure file	4.2.2
DO	Initiates execution of Procedure file.	Procedure file	4.1.3
DROP	Deletes all lines of all procs.	Directives	3.2.8
END	Returns user to proc zero.	Procedure file	4.1.2
FIND	Locates the first specified line and sets ? .	Data manipulation operations	3.1.2.11
FIRST	Modifies search commands to terminate after the first line containing the search-string.	Line and column range specifications	3.1.1.5
FSTATUS (file-parameter)	Transfers file information into the EDT work space.	Directives	3.2.11
GOTO	Transfers control within Procedure file.	Procedure file	4.3.1
HALT	Returns control to the monitor. Contents of the user's EDT work space are not destroyed.	Directives	3.2.1

(continued)

Table A-1 EDT Commands *(contd)*

Command	Function	Functional Category	Section
HELP	Displays additional information about the previous user error.	Directives	3.2.2
IF	Conditionally execute an EDT instruction.	Procedure file	4.3.2
INSERT (change-string)	Inserts the specified search-string at the column specified in the line-range. The contents of the existing line are shifted to make space for the insertion.	Data manipulation operations	3.1.2.4
LOWER	Specifies that both upper and lower case input will be entered from the terminal.	Directives	3.2.6
MOVE	Duplicates a line or range of lines and then deletes the original lines.	Data manipulation operations	3.1.2.5
NOP	Dummy command for line numbers and comments	Procedure file	4.3.3
NOT	Reverse sense of searches.	Line and column range specifications	3.1.1.10
NUMBER (sequence string)	Inserts sequence strings in lines entered from the terminal.	Data manipulation operations	3.1.2.12
ON (line-range)	Defines the line-range which limits data manipulation commands.	Line and column range specifications	3.1.1.9
PRINT	Displays all specified lines on the terminal.	Data manipulation operations	3.1.2.6
PROC	Change current Procedure file.	Procedure file	4.1.1
READ (file-parameter)	Transfers all or selected portions of an OS/3 Library file to the user's area.	Data manipulation operations	3.1.2.7
RSP	Terminates EDT; transfers its work space to the Remote Spool Processor.	Directives	3.2.10
SEQUENCE (sequence string)	Stores sequence numbers in specific columns of selected lines.	Data manipulation operations	3.1.2.8
SET PAGE=mm, LINE=ll, TABS=n1,n2,n3,n4,n5, n6,n7,n8 CHAR=x	Defines user tab parameters and terminal characteristics.	Directives	3.2.3

(continued)

Table A-1 EDT Commands *(contd)*

Command	Function	Functional Category	Section
SYSTEM [(BEM command)]	Returns control to BEM or executes a BEM command.	Directives	3.2.9
TO (change-string)	Specifies the character string to be substituted in text replacement.	Data manipulation operations	3.1.1.6
TO (copy-to location)	Specifies the output location for COPY and MOVE commands.	Data manipulation operations	3.1.1.7
TYPE	Displays the user's message on the 90/30 operator's console.	Directives	3.2.5
UPDATE	Displays selected lines on the user's terminal for modification. Lines which are retransmitted after updating will replace the existing line in the work area.	Data manipulation operations	3.1.2.9
UPPER	Specifies that lowercase input is to be translated to all uppercase.	Directives	3.2.7
WRITE (file-parameters)	Transfers selected lines of the user's work area to a permanent OS/3 Library file.	Data manipulation operations	3.1.2.10
(search-string)	Character string which data manipulation commands must find before operating on a given line.	Line and column range specifications	3.1.1.8
(line-range)	The line range on which data manipulation commands operate.	Line and column range specifications	3.1.1.9
@(line-number)	Sets the current line number and increment.	Directives	3.2.4

APPENDIX B OPERAND DEFINITIONS

Operand	Definition
(line-number)	A line number of the form nnnn.nnnn, not equal to zero.
(line-range)	Specifies a sequence of line numbers of the form a-b, where a,b, are line-numbers or a,b, as a set of lines or a combination of sets and ranges. An ampersand (&) specifies the entire work area.
(search-string)	A character string enclosed in apostrophes (') or quotes ("), or a hexadecimal string enclosed in apostrophes or quotes and preceded by an X. When apostrophes are used any occurrence of the string is searched for. When quotes are used only those occurrences bounded by delimiters are searched.
(change-string)	String to replace search string or to be inserted. Character string must be enclosed in apostrophes ('...'), hex string by X and apostrophes (X',,,').
(column-number)	One or two digit number specifying column number in a line.
(column-range)	Sequence of single or inclusive column numbers of the form a-b,c where a,b,c are column numbers.
(file-parameters)	ELT,FILENAME (PASSWORD),VSN,TYPE where: ELT is the name of the desired element. FILENAME is the name of the file containing the element. PASSWORD is the read or write password which may be required by the catalogue entry. VSN is the volume serial number of the disk pack on which the file resides. This may be omitted if the file has been catalogued. TYPE is the module's element type (S, P, M). If omitted, source is assumed.
(increment)	One to five-digit decimal number to be used as increment by the SEQUENCE command.
(sequence-string)	String to be used as initial value for SEQUENCE command. Of same form as change-string except rightmost digits must be numeric to be incremented, or an asterisk (*).
(proc-number)	Single digit integer (0-9).
(EDT-variable)	Symbol Gn or #Gn where n is a single digit integer (0-9).
(BEM command)	Any valid BEM command to be executed. (See Section 6.)

APPENDIX C SYSTEM ERROR MESSAGES

Error messages appropriate for an interactive environment are short and self-explanatory. For additional information about an error message, the user is directed to the HELP command. The messages are categorized on a functional basis and are listed in the following table. Possible causes of each error and suggested procedures to follow in response to the error are also included. The designation to the right of the message identifies the BEM component which identified the error.

A SUBSTATEMENT OCCURRED BEFORE AN END *BASIC*

Subprograms must occur after the main program. This means that they must be placed immediately after the END statement, or after another subprogram's SUBEND statement.

ACCESS TO PROGRAM NOT PERMITTED FOR USER ID *BEM*

The account currently in use does not permit its users to execute the selected function or program. Use another account, or contact the system administrator to change the account's restriction.

ACCESS TO QUEUE TYPE NOT PERMITTED *RSP*

Access to the queue type (PRINT, PUNCH, READER, LOG, JCS, RBPPU, RBPPR) is not permitted for one of two reasons. The OS/3 Supervisor has not been generated to include the appropriate level of support for the queue, or the BEM job control includes parameters to restrict access to that queue. Consult the system administrator to have the access type changed.

ACCESS TO SYSRES FILES NOT PERMITTED *LIBRARY*

The account currently in use does not permit its users to access files on the OS/3 system pack. Use another account or contact the system administrator to change the account's restriction.

ACTIVE SUBROUTINES EXCEED 16 LEVELS *BASIC*

A maximum of 16 levels of subprogram calls may be issued. Investigate for a possible program loop, or a recursive subprogram call.

ALLOCATE FORMAT ERROR *BEM*

The ALLOCATE command has been entered incorrectly. The format of this command is:

/ALLOCATE *type,file,vol,*[SIZE=*n*] [,INC=*n*] $\left[,\text{INIT}=\begin{Bmatrix}\text{YES}\\ \text{NO}\end{Bmatrix}\right]$

ARGUMENT TOO LARGE FOR EXP(X) FUNCTION *BASIC*

A value has been used with the exponential function which will produce a result greater than the largest number that the 90/30 is capable of handling. The maximum permissible value for the EXP argument is approximately 174.6.

ARRAY SUBSCRIPT OUT OF RANGE *BASIC*

An array subscript, which is out of the range specified by the dimension statement has been detected. The subscript is either less than zero, or greater than the upper limit in the dimension statement. If no dimension statement has been used, the upper limit is 10.

ATTEMPTED TO RESET FILE BEYOND EOF OR NEGATIVE *BASIC*

The RESET statement may not reposition the file past the end of the file pointer. The record number specified must be positive.

ATTEMPT TO TEST END OR MORE ON RANDOM FILE *BASIC*

The IF-END or IF-MORE formats may only be used against TERMINAL format files. Check the file type referenced by this statement.

BAD FORMAT — TRY AGAIN *RSP*

The user has transmitted something other than the preformatted parameter table. This may also be the result of using RSP with a UNISCOPE without the protected fields hardware option.

BASIC EDITING COMMAND UNRECOGNIZABLE *BASIC*

Either an invalid command has been entered, or a BASIC statement has been entered without a valid line number. Valid commands are:

OLD	NEW	SAVE
RUN	PRINT	HELP
BYE	DELETE	LIST
SYSTEM	RUNOLD	

BASIC FILE NOT OPEN OR NO DATA STATEMENTS *BASIC*

The channel number referenced by the flagged statement has not been opened by a FILE statement. Check the channel-setter for a valid file, or issue a FILE statement for the channel to be used. This error can also result when READ statements are issued and no DATA statements are present.

BASIC SOURCE LINES OUT OF ORDER *BASIC*

The lines of source in a BASIC program read in by a RUNOLD or CHAIN statement are not in order by line number. This is mandatory. Do an OLD against the program and then SAVE it.

BEM POINTERS DO NOT AGREE WITH WORKSPACE *WORKSPACE*

The workspace access routines have detected a problem with the in-core and disk pointers. This could have been caused by a previous I/O error, or a modification of the disk by an external source. If the error persists, the user may be forced to halt the current program and reexecute it.

BULLETIN LOCKED — RETRY LATER *BEM*

The bulletin cannot be updated because another user is currently accessing it. Wait until the other user finishes and retry the command. The system administrator should discourage the updating of the bulletin by multiple users.

CHAIN ERROR — INVALID NAME OR PASSING BAD FILE *BASIC*

There are two possible causes for this error. The Library element specified in the CHAIN statement does not exist, or one of the channel numbers of files to be passed to the next program segment is invalid.

CHANGE ERROR *BASIC*

The CHANGE operation specified by the flagged statement is not valid. Possible causes of this error are an invalid vector or vector size, invalid BIT expression, or invalid string result, or invalid value encountered during conversion.

CHANNEL NUMBER INVALID IN FILE STATEMENT *BASIC*

The channel-setter used in the FILE statement results in a channel number which is not in the range 1 to 4095. Channel zero cannot be defined by a FILE statement.

COMMAND CANNOT BE USED AT THIS TIME *BEM*

A PRINT, PUNCH, DELETE, or FSTATUS command was issued while an active program had been interrupted. The active program was accessing a file at the time of interruption. Allow the interrupted command to complete (RESUME) and then retry the command.

COMMAND KEYWORD OMITTED *EDT*

An operand has been entered for which there is no command function. For example, the file parameters have been used without the specification READ or WRITE.

COMMAND TERMINATED *EDT*

The EDT command which was active when the user issued a /INTR or DISCONTINUE command has been terminated. Informational message only.

CONTINUE? (Y OR N) *ALL*

BEM has displayed a full screen or page and has additional output for the terminal. When ready, the user may respond with a Y to see additional displays, or an N to terminate the display and the command. A response other than Y or N will result in the CONTINUE message being displayed again.

COPY WITH NUMBER OPTION INVALID *EDT*

The COPY command may not be used with the NUMBER command.

DEF MUST PRECEDE REFERENCE IF LOCALS ARE USED *BASIC*

When local variables are used in a multiline user function, the definition must occur at a lower numbered line than the first reference to that function. Move the function definition and rerun.

DESEQ OPTION ONLY ALLOWED WITH READ *EDT*

The DESEQUENCE option is only meaningful when used with the EDT READ command; in all other cases its use is treated as an error.

DEVICE UNAVAILABLE AT THIS TIME *BEM*

The printer or punch is not configured and may not be used. Contact the system administrator to have the printer or punch configured.

DIMENSIONS INCONSISTENT IN SUB CALL *BASIC*

The type of variables used in the SUB and CALL lines differ. Either a scalar variable was used where an array was expected, or the number of subscripts on the SUB and CALL lines differ.

DISPLAY COMMAND PARAMETER ERROR *BEM*

The DISPLAY command has been entered incorrectly. Valid options are:

$$\text{/DISPLAY} \quad \left\{ \begin{matrix} \text{JOBS} \\ \text{VOLUMES} \end{matrix} \right\}$$

DIVISION BY ZERO, EXECUTION CONTINUES *BASIC*

The program has attempted a division by zero. The algebraic result of division by zero is undefined; however, execution continues using a high value.

EDT VARIABLE AREA NOT AVAILABLE *EDT*

There is currently insufficient storage available to use EDT variables.

ELEMENT/GROUP NOT FOUND *BEM*

The element or group specified in the DELETE command could not be found. Check the spelling of the name and check the names in the file via FSTATUS.

ELEMENT IS NOT IN THE LIBRARY FILE *LIBRARY*

The program requested by the command is not in the file specified. Check the spelling of the program name and verify that the program is on the file. Also, be sure the correct module type has been used (P for PROCs).

ELEMENT NUMBER DOES NOT EXIST, RE-ENTER ⊳ *BEM*

The user did not select one of the numbers listed by the RECOVER command. Only those elements identified with a number in the left margin may be recovered. Reenter the correct number and the new module name.

END OF FILE ON INPUT OR LINPUT *BASIC*

The program has issued an INPUT or LINPUT statement which attempted to read more records than were in the file. Investigate the program logic to determine why too many records are being read.

END STATEMENT IS MISSING OR MISPLACED *BASIC*

All BASIC programs must have an END statement as the last line. Insert an END statement and rerun.

ENTER ELEMENT NUMBER, NEWNAME OR "STOP"⊳ *BEM*

The RECOVER command has presented a list of elements which could be recovered. Select one by specifying its number, and a new name for it. Other possible responses at this point are STOP to terminate the command, or HELP to obtain additional information.

ENTER FILE NAME *BASIC*

The user has entered a SAVE, OLD, or RUNOLD statement without specifying a file name. Supply the name in response to this message.

ENTER FUNCTION NOT CONFIGURED *BEM*

The system administrator has not elected to make the ENTER command available at your site. Contact the administrator to have the function installed. This error may also be the result of not having OS/3 Spooling configured, or not having any spooled Input Readers.

ERROR IN READING CARDS/ENTER STREAM, USER CANCELLED *BEM*

A fatal I/O error has occurred while reading cards from a batch stream or enter file. The batch is discarded and the user is cancelled.

ERROR IN READING SCRATCH SPACE *WORKSPACE*

An I/O error has occurred while reading from the work area. Retry input or investigate for possible hardware problem.

ERROR IN READING SCRATCH SPACE INDEX *WORKSPACE*

An I/O error has occurred while reading the work area index. Retry input or investigate for possible hardware problem.

ERROR IN SOURCE — RESEQUENCE TERMINATED *BASIC*

One or more of the source statements read in by an OLD command with errors have not been corrected. Only valid programs in the workspace may be resequenced. This error indicates that there is at least one statement which is not syntactically correct.

ERROR ON READ FROM FILE (INVALID NUMBER) *BASIC*

A READ statement attempted to read a numeric variable. The record which was read did not contain numeric data.

ERROR PROCESSING USER FILE LABELS *FILES*

The file being accessed contains user file labels. These cannot be processed by BEM.

ERROR WHILE WRITING INTO SCRATCH SPACE *WORKSPACE*

An I/O error has occurred while writing to the work area. Retry input or investigate for possible hardware problem.

EXPONENT OVERFLOW, EXECUTION CONTINUES *BASIC*

The result (or intermediate result) of a computation has exceeded the largest number the 90/30 is capable of handling. This number is approximately 10^{75}. Machine infinity is supplied and execution continued.

EXPONENT UNDERFLOW, EXECUTION CONTINUES *BASIC*

The result (or intermediate result) of a computation is less than the smallest number the 90/30 is capable of handling. The number is approximately 10^{-78}. Zero is supplied and execution continued.

EXPONENTIATION ERROR *BASIC*

Invalid operands were used with the A**B or A↑B function. This error can occur if "A" is negative and "B" is not an integer between 1 and 15 or −1 and −15.

EXPRESSION OUT OF COMPUTED GOTO RANGE *BASIC*

The calculated expression is not a valid number for this computed GOTO. It is either too large or nonpositive. The count of line numbers in the statement determines the largest value the expression may have.

FILE ACCESS HAS BEEN TERMINATED BY USER *LIBRARY*

This indicates that a file access has been terminated when the user did not wish to wait on a FILE IS IN USE message.

FILE ALREADY EXISTS ON VOLUME *BEM*

The user is attempting to allocate a file which already exists on the specified volume.

FILE DOES NOT HAVE VALID "ENDLIB" *LIBRARY*

While searching the directory of the file, BEM could not find the ENDLIB marker. The file's integrity is in question. A possible solution would be to copy all elements to another file, then scratch and rebuild the original file.

FILE IS EMPTY — ENDLIB MISSING *LIBRARY*

The user has attempted to access an empty library file. Initialize the file with the Librarian in order to use it with BEM.

FILE IS IN USE, PLEASE WAIT *LIBRARY/FILES*

Another user is accessing the file. After his command completes, yours will begin. If you don't wish to wait, interrupt the system.

FILE IS NOT AN OS/3 LIBRARY FILE *LIBRARY*

The file specified by the command is not a library file, or has not been initialized by the librarian. Have the system administrator prepare the file, and be sure you are using the correct file.

FILE PARAMETERS DO NOT FOLLOW "FSTATUS" *BEM*

The FSTATUS command requires file parameters in the format:

filename (password), volume

FILE PARAMETER FORMAT ERROR *LIBRARY/FILES*

The file parameters given for a file-access function are not valid. The maximum length for each parameter is: name, 8; filename, 44; password, 6; volume, 6. If a module type has been supplied, it must be S, P, or M.

FILE REQUESTED IS NOT ON DISC VOLUME *LIBRARY/FILES*

The filename requested is not on the volume specified. Check the spelling of the filename or verify that the file is on the volume.

FILE STATEMENT INVALID FOR # 0 *BASIC*

The channel-setter specified with the FILE statement results in a value of zero. Channel zero, the terminal, cannot be defined by a FILE statement.

"FNEND" FOUND WITHOUT FUNCTION DEFINITION *BASIC*

The FNEND statement was detected, but it was not at the end of a function. Remove the statement or place it in the correct location and rerun.

"FNEND" STATEMENT MISSING *BASIC*

A user-defined multiline function exists in the program without a closing FNEND statement. Locate the function and insert the FNEND statement.

FUNCTION ASSIGN DOES NOT MATCH FUNCTION NAME *BASIC*

The name of the function being assigned differs from the name of the function in which it appears. Only the function being defined may be assigned a value.

FUNCTION ASSIGNMENT MUST APPEAR WITHIN FUNCTION *BASIC*

A value must be assigned to a multiline function before the FNEND statement. The function value may not be assigned outside the body of the function.

FUNCTION DEF MUST PRECEDE USE IN "CALL" *BASIC*

In order for a user function to be passed to a subprogram, it must be defined. Move the definition into lower-numbered lines before the CALL.

FUNCTION DEFINITION WITHIN A FUNCTION *BASIC*

BASIC has detected a function within the body of another function definition. Check for a missing FNEND statement or restructure the function.

FUNCTION EXPECTED IN CALL OR SUB LINE *BASIC*

A previous CALL statement passed a function reference. This CALL did not pass a function. The parameter types must be the same. Resolve the conflict and rerun the program.

FUNCTION HAS NOT BEEN DEFINED *BASIC*

The function referenced on the line in error has not been defined. Define the function or remove the reference to it and rerun.

GIVEN LINE EXCEEDS 80 CHARACTERS WHEN RESEQUENCED *BASIC*

The line shown, when resequenced, is larger than 80 characters. This is an informational message, in that the complete resequenced line is written out, and can be modified by EDT, but if the program is later read in by BASIC, it will be flagged with an error for being over 80 characters in length.

GOTO INTO OR OUT OF FUNCTION DEFINITION *BASIC*

A function may not reference program lines which do not occur within the body of the function, nor may statements outside the function reference lines within the function body. This applies to GOTO, GOSUB, ON, and IF statements.

ICAM ERROR (INPUT TOO LONG) RETRY *BEM*

The last message sent from the terminal to BEM did not arrive correctly; retransmit it. Any input to BEM is limited to 128 characters in length. If this error is displayed while transmitting the RSP Spool file descriptor screen, it indicates the UNISCOPE being used does not have the required protected format feature.

ILLEGAL COMBINATION — "NOT" INVALID *EDT*

If the NOT option is specified, then a search-string must also be specified and a change-string must not be specified.

ILLEGAL COMBINATION OF COMMANDS *EDT*

Several command keywords have been entered which conflict. See Table 3-1 in UA-0141 for allowable combinations.

ILLEGAL "VAL" ARGUMENT *BASIC*

The string passed to the VAL function did not contain a valid number. The contents of the string must be either an integer or a decimal number in scientific notation. No extra characters may be prefixed or suffixed to the number.

INCORRECT NESTING OF FOR-NEXT STATEMENTS *BASIC*

A FOR or NEXT statement, which was not nested correctly, was detected. Possible causes are:

1. A FOR statement that has the same index as a previous FOR statement in the nest.
2. A NEXT statement that does not have the same index as the FOR statement immediately preceding it.
3. A NEXT statement that does not follow any open FOR statement.

INCONSISTENT FORMAT IN "USING" STRING *BASIC*

The format field type does not match the type of variable being printed. Either a string was printed into a field beginning with $, + or −, or a number was printed into a field beginning with < or >.

INPUT DATA INCORRECT, RE-ENTER *BASIC*

The data entered for an input statement does not match the variable types required by the program. The entire line must be reentered. This error message could also be caused by too much or too little data in the input response.

INSERT ERROR (DUPLICATE OR INVALID CHANGE STRING) *EDT*

Either the keyword INSERT is preceded by a string, or it is not followed by one. The change string may also be invalid. See Section 3.1.2.4 or 3.1.1.6 of UA-0141.

INSUFFICIENT DATA TO READ *BASIC*

All DATA statements in the program have been used, yet the program attempted to request additional data.

INSUFFICIENT INFORMATION TO CREATE SPOOL FILE *RSP*

The minimum information required to create a Spool file was not specified on the spool descriptor screen. An input file requires either an LBL, or both a JOB NAME and an LFD.

INSUFFICIENT RESOURCES TO LOGON *BEM*

There is not enough memory or user tasks to allow another user to log on to the system. The user should wait until another user has released storage or logged off.

INTERNAL ERROR IN LIBRARY ACCESS ROUTINE *LIBRARY*

The library access routine within BEM has detected a logic error. Take a dump as soon as possible, save all relevant data, and consult your Sperry Univac representative.

INTERNAL ERROR IN RESEQUENCE ROUTINE *BASIC*

A condition which should not normally exist has been detected by the resequence routines. Collect all relevant data, obtain a memory dump, and contact your local Sperry Univac customer representative.

INTERNAL ERROR IN WORKSPACE *WORKSPACE*

An internal error has been detected by the workspace access routines in BEM. If the error persists the user may be forced to halt the current program and reexecute it.

INTERRUPTED: (C)ONT,(D)ISCONT,(S)YSTEM ⊳ *BEM*

This message indicates that the user has interrupted BEM by means of the MESSAGE-WAITING key on a UNISCOPE terminal, or the BREAK key on a hardcopy terminal. The user has three options: C- will continue the interrupted operation; D- will discontinue the current operation and return to command mode; S- will temporarily suspend the current operation and allow the user to enter BEM commands; when the user wishes to resume the current operation, the /RESUME command is used.

INVALID @(LABEL) -- MISSING PAREN *EDT*

An open parenthesis was found to start a label, but there is no closing parenthesis.

INVALID @SET COMMAND *EDT*

An @SET command has been used with an invalid keyword parameter. The only valid keywords for use with SET command are PAGE, LINE, TABS, and CHAR. See Section 3.2.3 of UA-0141.

INVALID ASSIGN STATEMENT *EDT*

An ASSIGN statement must be of the form:

@ASSIGN G*n* = *expression*

INVALID BLOCKSIZE OR RECORD SIZE *FILES*

BEM cannot process the file due to a conflict with the block or record size for this file. If the file already exists, check that the block size or record size is not zero, or greater than 65K.

INVALID BULLETIN OPTION — NOT READ/WRITE/DISPLAY *BEM*

There are only three valid bulletin functions which may be used with the BULLETIN command. These are READ, WRITE, and DISPLAY. Correct the command and retry it.

INVALID BY PARAMETER USAGE *EDT*

The BY specification has been used without the SEQUENCE command, or the form of the parameter is not valid. See Section 3.1.1.2 of UA-0141.

INVALID CHANNEL SET EXPRESSION *BASIC*

The channel-setter in the flagged statement resulted in a number less than zero, or greater than 4095. Channel numbers must be between 0 and 4095.

INVALID COLUMN IN TAB COMMAND *EDT*

One of the column numbers used in a TAB command is not between 1 and 128. See Section 3.2.3 of UA-0141.

INVALID COLUMN RANGE *EDT*

The column range specified in a replacement expression is invalid. It must be of the form:

```
n:i-j
```

where $1 \leq i \leq j \leq 128$

INVALID DO OPTION *EDT*

The DO statement has the format:

```
@DO n [P]
```

where n is an integer (1-9) and the optional P specifies that each command is to be printed.

INVALID EDT VARIABLE (#Gn) *EDT*

A single number sign (#) is assumed to designate an EDT general variable. This must be followed by the letter G and a digit in the range 0-9. If a number sign is needed in the command, enter two number signs (##).

INVALID ELEMENT TYPE *BEM*

The element type used with the /DELETE command must be one of the following:

S-source	P-proc	M-macro
O-object	L-load	G-group

INVALID EXPONENT FIELD IN USING STRING *BASIC*

An exponent field must consist of exactly five up-arrows (↑) and cannot be followed by a place holder #. Correct program and rerun.

INVALID FIELD DESCRIPTOR, EXPECTING <, > *BASIC*

The user program attempted to print a string variable with a numeric format. Correct the program and rerun.

INVALID FIELD DESCRIPTOR, EXPECTING $,+,— *BASIC*

The user program attempted to print a numeric variable with a string format. Correct the program and rerun.

INVALID FORMAT FOR LOGON COMMAND *BEM*

The LOGON command has been entered incorrectly. It must begin with the word LOGON, and is followed by one to three fields of up to four characters each. Check that none of the fields are too long, and that there is nothing entered after the third field.

INVALID ID, ACCOUNT, PASSWORD FOR LOGON *BEM*

An unlisted id, account, and password combination has been entered; thus the user has been denied access to the system. If the fields have been entered correctly, then the account may have been removed from the system. Contact the system administrator to have the account created.

INVALID IF STATEMENT *EDT*

An IF statement has the following format:

@IF *expression op expression COMMAND*

@IF $\left\{\begin{matrix}.T.\\.F.\end{matrix}\right\}$ *COMMAND*

INVALID KEY LENGTH *FILES*

Files containing keys cannot be processed by BASIC.

INVALID LINE RANGE *BASIC*

Valid line ranges consist of single line numbers (a,b) or ranges of lines (a-b). A line number consists of a decimal number in the range 1-99,999.

INVALID LINE SET COMMAND *EDT*

An at sign (@) alone has been entered, or the line number with the line set command is not valid. See Section 3.2.4 of UA-0141.

INVALID OR ZERO LINE NUMBER *EDT*

A line number in an EDT variable expression must be in the form nnnn.nnnn, and must not be zero.

INVALID MAJOR FRAME COMMAND *RSP*

The user has entered a command other than one of those shown on the screen. A new screen will be presented. Valid commands are:

BREAK	END	DISPLAY	RETRIEVE
RELEASE	BUILD	DELETE	HELP
CLEAR	READ	WRITE	SCREEN
TYPE	UPPER	LOWER	SYSTEM

INVALID MARGIN SIZE *BASIC*

The margin expression specified on the flagged statement resulted in a number less than zero, or greater than 4095. This error could also have resulted from attempting to set the size of the margin greater than the limit for the file type.

INVALID NUMBER PARAMETER *EDT*

The number parameter must follow the NUMBER command, must be a valid change string, and must terminate with at least 1 but not more than 15 numeric characters. The parameter must be enclosed in apostrophes and any "or" characters in the string must be entered twice.

INVALID OR DUPLICATE CHANGE STRING *EDT*

The change-string used is not valid or two change-strings have been entered. Change-strings must begin and end with apostrophes. See Section 3.1.1.6 of UA-0141.

INVALID OR DUPLICATE COLUMN RANGE *EDT*

The column range entered is not valid, due to incorrect format, or two column ranges have been entered. See Section 3.1.1.3 of UA-0141.

INVALID OR DUPLICATE COPY-TO LOCATION *EDT*

Either two copy-to locations have been used (i.e., @ COPY 1-10 TO 11 TO 22) or the one given is not valid. The number following the word TO must be a valid line number. See Section 3.1.1.7 of UA-0141.

INVALID OR DUPLICATE LINE RANGE *EDT*

The line range entered is not valid due to incorrect format or two line ranges have been entered. See Section 3.1.1.9 of UA-0141.

INVALID OR DUPLICATE SEARCH STRING *EDT*

The search-string used is not valid or two search-strings have been entered. Strings must begin and end with quotes or apostrophes. See Section 3.1.1.8 of UA-0141.

INVALID PROC GROUP NUMBER *EDT*

The PROC group number must be a single digit integer in the range 1-9.

INVALID RESPONSE, ENTER NUMBER, NEWNAME▷ *BEM*

The user's response to the last query was incorrect. A non-zero number must be entered first, followed immediately by a comma and then the module name. No intervening spaces are permitted.

INVALID SCREEN ROLL COMMAND *RSP*

The user has entered a command other than one of those shown on the top of the screen. A new screen will be presented. Valid commands are:

CMD	UP	DOWN
RIGHT	LEFT	DELETE
INSERT	UPDATE	REFRESH

INVALID SEARCH COMMAND *RSP*

The search command issued to RSP is not correct. It consists of a search-string and an optional column range. The search-string must begin and end with an apostrophe. A column range is a single number, or two numbers separated by a hyphen. The number must be between 1 and 256.

INVALID SEARCH STRING *BASIC*

A search-string consists of any character string enclosed in quotation marks. If a quote appears in the string, it must be typed as " ".

INVALID SUBSTRING EXPRESSION *EDT*

A substring of an EDT variable is written as a starting position (s) and a length (l) enclosed in parentheses--(s,l).

where $1 \leqslant s \leqslant 50$ and $s + l \leqslant 51$

INVALID TAB EXPRESSION FOR PRINTING *BASIC*

The argument of the TAB function was less than one.

INVALID TRIMMER IN MATRIX STATEMENT *BASIC*

Either the trimmer specified did not result in a positive number, or the resultant array would require more storage than the original array.

INVALID VARIABLE EXPRESSION *EDT*

An EDT variable expression must be one of:

STRING — 'ABC'
VARIABLE — Gn
NUMERIC EXPRESSION — n + m, n − m, n
LINE/COLUMN RANGE — n:i−j

I/O AREA COULD NOT BE LOCATED, RETRY *LIBRARY*

An I/O area for the library function could not be acquired by BEM. Wait a few minutes and retry. If the problem persists, contact the system administrator to have the system memory partition enlarged.

I/O ERROR ACCESSING MESSAGE INDEX *BEM*

An I/O error has occurred while writing to the bulletin file. Only part of the bulletin is now valid. The DISPLAY option should be used to determine that status of the bulletin, and the WRITE option may then be retried.

I/O ERROR ON WRITE TO FILE *FILES*

An I/O error has occurred while writing the data management file. Investigate for possible hardware problem or retry the program.

I/O ERROR WHILE ACCESSING V.T.O.C. *LIBRARY/FILES*

An I/O error has occurred while accessing the VTOC for the disk volume specified. Retry or investigate for possible hardware problem.

I/O ERROR WHILE READING CATALOG *LIBRARY/FILES*

An I/O error has occurred while reading the catalog. Retry or investigate for possible hardware problem.

I/O ERROR WHILE READING LIBRARY FILE *LIBRARY*

An I/O error has occurred while reading the library file. Part of the program may be missing. Retry or investigate for possible hardware problem.

I/O ERROR WHILE WRITING LIBRARY FILE *LIBRARY*

An I/O error has occurred while writing the library file. The program has not been saved. Retry the command or investigate for possible hardware problem.

LIBRARY FILE FULL, ELEMENT NOT ADDED *LIBRARY*

The library file has been filled and there is not enough room to write out the program. The old version, if any, is left intact. Have the file expanded or its contents compressed.

LIMIT OF 4 "LIBRARY" STATEMENTS EXCEEDED *BASIC*

BASIC will search at most four libraries for subprograms, the program has attempted to use more than four.

LOADER AT LINE xxxxx *BASIC*

When the error was detected, the BASIC compiler was at the line number given by xxxxx. This message is displayed in conjunction with another error message.

LOG OF A NON-POSITIVE NUMBER UNDEFINED *BASIC*

The LOG function has encountered a nonpositive argument. The logarithm of a nonpositive number is undefined, thus execution is cancelled.

MATRIX DIMENSIONS ARE INCORRECT FOR FUNCTION *BASIC*

The row or column dimension of the matrices in the matrix statement is incorrect. Check DIM statement for the matrices in question.

MISSING FILE PARAMETER *EDT*

A READ or WRITE command has been entered, but the file parameters do not immediately follow the command keyword. Correct and retry.

MISSING FILE PARAMETER *BEM*

File parameters must immediately follow the DELETE, PRINT, or PUNCH keyword and be in the format:

element, filename (password), volume, type

MODULE NOT OVERWRITTEN, COMMAND TERMINATED *LIBRARY*

This is a confirmation message informing the user that the WRITE command was not executed. It results from a NO answer to the OVERWRITE question.

MORE THAN 29 FILES OPEN *BASIC*

BASIC does not support the concurrent use of more than 29 temporary and library files per user. This program has exceeded the limit.

MORE THAN 4 CHARACTERS IN LABEL *EDT*

EDT statement labels may contain no more than four characters. Correct the proc by using shorter labels.

MUST BE PRIVILEGED FOR BULL READ/WRITE *BEM*

Only privileged users may read or write the BEM bulletin. Normally, only the system administrator will have a privileged status in the accounting file.

NEW NAME ALREADY EXISTS, RE-ENTER ▷ *BEM*

An element with the specified name already exists in the file. Select another name and retry the response.

NO DISK SCRATCH SPACE AVAILABLE *WORKSPACE*

All the disk cylinders available to BEM have been assigned, wait and retry or contact the system administrator.

NO FORMAT STRING DEFINED IN USING STRING *BASIC*

The user program attempted to print a variable using a format string that does not contain any valid format strings.

NO MEMORY AVAILABLE FOR FILE I/O BUFFERS *FILE*

A memory area to store a block buffer and DTF could not be allocated for your file. Retry later and contact the system administrator if the problem persists.

NO MEMORY AVAILABLE FOR WORKSPACE BUFFERS *WORKSPACE*

An area of memory could not be acquired for I/O buffers. Retry command when memory becomes available. If problem persists, contact system administrator to have the memory partition size increased.

NO PROC TO END *EDT*

The @END statement was issued while no proc was active.

NO SUCH LINE NUMBER FOR A GOTO OR GOSUB OR IF-THEN *BASIC*

The line number referenced in a GOTO, GOSUB, ON, or IF-THEN statement is not present in the program or function. Insert the required statement or remove the reference to it.

NOT A DATA MANAGEMENT FILE *FILES*

The file being accessed is not a valid sequential or direct access file. To be valid, it must have type SQ or DA and contain a single partition.

NOT ENOUGH MEMORY IS AVAILABLE TO LOAD *BEM*

Insufficient storage is available to load the function you are calling for. Wait and retry. If the problem persists, contact the system administrator to have the memory partition size increased.

NOTHING HAS BEEN FOUND TO RECOVER *BEM*

The RECOVER command has searched the library for deleted modules with the same name and type as specified in your command, but could not find any. This response may indicate that the library has been packed, or that you have recovered all deleted elements and there aren't any left to display.

NULL USING STRING NOT ALLOWED *BASIC*

The using string specified in the print statement is a null string. Define the variable and rerun.

NUMBER OF ARGUMENTS INCONSISTENT *BASIC*

The number and type of arguments passed in the CALL statement(s) do not agree with the number and type on the SUB line.

NUMBER OF PARAMS IN FUNCTION CALL INVALID *BASIC*

A maximum of 16 passed parameters and local variables may be specified on a function declaration line. Reduce the number of labels and rerun.

NUMBER OF SUBSCRIPTS FOR ARRAY INCORRECT *BASIC*

The variable that caused the error has been dimensioned with a different number of subscripts than were found in the reference to it.

OPERATION NOT PERMITTED TO FILE *BASIC*

The operation to be performed against the file conflicts with the file type.

OS/3 ALLOCATE ERROR *BEM*

This message is returned when the ALLOCATE command receives an error status from the supervisor when trying to allocate the file. It may indicate that there is insufficient space on the disk volume.

OUT OF MEMORY — RETRY (Y OR N) *ALL*

One of the internal routines within BEM has attempted to acquire additional storage on a temporary basis. No storage was available. The user may wait for storage to become available and reply Y, or may terminate the current program by replying N. If the problem persists, contact the system administrator to have the memory partition size increased.

OVERFLOW ON VARIABLE SUBSTITUTIONS — TRUNCATED *EDT*

When variables in a command line were replaced, the new line exceeded 80 characters. The truncated command was processed.

OVERWRITE? (YES OR NO) *LIBRARY*

The program to be written out by the command already exists on the file. A reply of YES will overwrite the previous version with the new one. A reply of NO will terminate the command.

PAGE/LINE SIZE INVALID *EDT*

The page or line sizes are not within the correct range. PAGE must be between 1 and 255, LINE must be between 1 and 128.

PARAMETER TYPE MIS-MATCH *BASIC*

The type of a parameter passed to a function/subprogram conflicts with the type defined for the function subprogram. For example, a string was passed when a numeric value was expected or a numeric value was passed when a string was expected. Compare the line in error and the definition; correct the discrepancy.

PASSWORD IS INVALID FOR FILE *LIBRARY/FILE*

The password used does not match the one cataloged for the file. Another cause of this error could be failure to specify a password with the file-access command. The user is denied access to the file in either case.

PAUSED AT xxxxx CONTINUE (Y or N) *BASIC*

A PAUSE statement has been encountered at the line number given by *xxxxx*. Answer YES to continue execution; answer NO to terminate the program.

PLEASE LOGON *BEM*

The user's terminal has not been joined to the BEM system. Follow log-on procedures given in Section 2 of UA-0139.

PRINT TO FILE > MARGIN SIZE *BASIC*

The program attempted to print a string, number or USING string with a length greater than the current margin setting. Change the margin size, or reduce the length of the expression printed.

PRINT/PUNCH I/O ERROR *BEM*

A hardware I/O error has been encountered on the printer or punch. Retry the command. If the problem persists, investigate a possible hardware error.

PRINTER/PUNCH IS IN USE, PLEASE WAIT *BEM*

Another user is using the printer or the punch. Your command will be completed after the other command completes. If you do not wish to wait, interrupt the system.

PROGRAM CANNOT BE RESUMED *BEM*

The user tried to resume a program when no program had been loaded. A RESUME command is only effective when the user has interrupted an active program and wishes to return to it.

PROGRAM COULD NOT BE FOUND *BEM*

The program to be executed via an EXECUTE command could not be found. Only EDT, RSP, and BASIC may be loaded under level 4.0.

PROGRAM NOT INCLUDED IN CONFIGURATION *BEM*

The system administrator has not elected to provide the program you have requested.

REFERENCE TO ACTIVE PROC *EDT*

The user has attempted the DO option on a currently active proc, or has attempted to enter the current proc with an @PROC command.

REFERENCED SUBROUTINE NOT FOUND IN LIBRARIES *BASIC*

All user-specified libraries have been searched, but the subprogram listed in the error message could not be found. Execution is inhibited.

RENAME ERROR *BASIC*

The string-expression used to supply the new file name does not contain a valid file parameter or temporary file name. This error may also be the result of attempting to RENAME a data management file.

REQUESTED RECORD NOT FOUND IN DATA FILE *FILES*

BASIC was attempting to read a record which does not exist in the data management file. Probably due to a hardware error. If the problem persists, consult your Sperry Univac representative.

RETURN WITHOUT MATCHING GOSUB CALL *BASIC*

The program has attempted to return from a subroutine that was not called by a GOSUB statement.

ROLL OPTION VALID ONLY AT UNISCOPE TERMINALS *BEM*

The /SCREEN ROLL or /SCREEN COP options cannot be used with a hardcopy terminal. The command is ignored.

RSP AVAILABLE ONLY AT UNISCOPE TERMINALS *RSP*

RSP cannot be used at a hardcopy terminal. Move to a UNISCOPE terminal and reexecute RSP.

RSP/EDT MUST BE LOADED TO USE/BULLETIN *BEM*

The BULLETIN READ or WRITE commands can only be issued while EDT or RSP is loaded (use @SY BULLETIN. . .) since there is no workspace unless one of these is active.

SAME MATRIX APPEARS ON BOTH SIDES OF EQUAL SIGN *BASIC*

The same matrix may be referenced on both sides of the equal sign in a MAT statement, a new matrix must be generated.

SAT ERROR INITIALIZING FILE *LIBRARY*

The INIT=YES option has been selected in the command, and the file could not be initialized. This could be due to a hardware error, or an attempt to initialize a non-SAT file.

SCRATCH AREA IS FULL, TEXT NOT ADDED *WORKSPACE*

The program you are using has tried to acquire an additional unit of disk space and could not do so. The last image entered has been lost. Wait and retry or contact the system operator.

SCRATCH ERROR *BEM*

The file specified in the command could not be scratched. This may be a result of an error status being returned from the supervisor, or could have been caused by an attempt to scratch a file which should not be scratched (a system file for example).

SCREEN COMMAND FORMAT ERROR *BEM*

The SCREEN command has been entered incorrectly. Valid options are:

$$/\text{SCREEN} \quad \begin{bmatrix}\text{ROLL}\\ \text{NOROLL}\end{bmatrix} \quad \begin{bmatrix}\text{COP}\\ \text{'NOCOP}\end{bmatrix} \quad \left[,\textit{height} \times \textit{width}\right]$$

The default is /SCREEN NOROLL,NOCOP,24×80.

SCREEN DIMENSIONS ARE INVALID FOR RSP *RSP*

RSP may only be used with UNISCOPE terminals; the only valid sizes for these terminals are 12 × 80, 16 × 64, 24 × 80, and 24 × 64. These are the only sizes which will be accepted.

SEARCH STRING NOT FOUND *RSP*

RSP has searched the workspace from the current location to the end, but could not find the string requested. Informational message only.

SEARCH STRING NOT FOUND IN LINE-RANGE *EDT*

The Editor has scanned all lines that the user's command has instructed it to, but did not find the string for which it was searching. This is caused by looking for a word or string which is not in the text. Informational message only.

SECOND DEFINITION OF AN ARRAY NOT ALLOWED *BASIC*

Two-dimension statements have been used to define the same variable. Remove one of the statements and rerun.

SECOND DEFINITION OF THE SAME FUNCTION *BASIC*

The same function has been defined twice within the program. Remove one definition and correct the program. Rerun.

SECOND DEFINITION OF SUB — DEFINITION IGNORED *BASIC*

Two subprograms with the same name have been encountered during the compilation process. The second subprogram will be ignored. The second subprogram may have been found in a library element as a result of a library search. This is a nonfatal error.

SEQUENCE PARAMETER ERROR *EDT*

The name or number used to sequence a module is not correct. This may be caused by using more than 16 numeric characters in the name or increment number. See Section 3.1.2.8 of UA-0141.

SET MARGIN FOR DMS FILE NOT AT RECORD 0 *BASIC*

A MARGIN statement was issued against a data management file while it still has data in it. The MARGIN statement may only be used when the file is empty.

SIMPLE VARIABLE INCONSISTENT WITH CALL *BASIC*

The CALL and SUB lines differ in the specification of a simple variable to be passed to the subprogram. Resolve the inconsistency and rerun.

SOFTWARE CHECK AT ee LLLLLL *BEM*

A software check has been detected by the Monitor. Please take a dump as soon as possible; save all relevant data, and consult your Sperry Univac representative.

SPECIFIED LINE NOT IN FILE *EDT*

The line specified in a replacement expression does not exist in the EDT work space.

SPOOL FILE NOT FOUND — COMMAND IGNORED *RSP*

The file described is not present in the Spool file. Check the spelling of the entries, and check that the correct queue name was specified. The spool element may not have been created yet.

SPOOL I/O ERROR R*U* *RSP*

An I/O error occurred while accessing the system Spool file. Respond R to retry; U to terminate the command. If invalid data has been retrieved, clear the workspace (CLEAR) and retrieve the file again.

SPOOL I/O ERROR WHILE ENTERING TASK *BEM*

The ENTER function has encountered a Spool file access error while writing the command element to the Spool file. If the error persists, contact the system administrator.

SQUARE ROOT OF A NEGATIVE NUMBER UNDEFINED *BASIC*

The SQR function has encountered a negative argument. The square root of a negative number is undefined, thus execution is cancelled.

START AND INCREMENT WILL EXCEED 99999 *BASIC*

The starting number and increment used with the RESEQUENCE command cannot be used as they are, because they would cause one of the new line numbers to exceed the maximum line number (99999) for OS/3 BASIC. Use a different start or increment and reissue the command.

STATEMENT FOLLOWING END/SUBEND NOT SUB/REM *BASIC*

The only permissible statements following an END statement are a REM statement or a SUB statement. Correct the program and rerun.

STATEMENT LABEL NOT IN FILE *EDT*

The label specified on the @GOTO statement could not be found in the EDT proc space.

STATUS COMMAND PARAMETER ERROR *BEM*

The operand of a STATUS command is incorrect. Allowable status commands are:

/STATUS	TERM
/STATUS	RESOURCE
/STATUS	

STOPPED AT xxxxx *BASIC*

A STOP statement has been encountered or an error detected at the line number given by xxxxx.

STRING EXCEEDS 4095 CHARACTERS *BASIC*

A string operation has produced a string with a length in excess of 4095 characters. The maximum number of characters permitted in a string is 4095.

SUB: FNX PRECEDES "CALL" *BASIC*

A SUB statement declaring a passed function cannot occur before the statement that calls it (and defines the function parameters). Relocate the subprogram so it occurs after at least one statement that calls it.

SUB NAME IS GREATER THAN 8 CHARACTERS *BASIC*

The name used on a CALL or SUB statement for a subprogram must be a string constant which is not longer than 8 characters. Correct the spelling of the name or shorten its length.

"SUBEND" OR "SUBEXIT" NOT IN A SUB *BASIC*

A SUBEND or SUBEXIT was encountered which was not in a subprogram. The SUBEND must be the last statement in a subprogram.

"SUBEXIT" NOT ALLOWED IN FUNCTION DEFINITION *BASIC*

A SUBEXIT statement was encountered within a multiline user function definition. It can only be issued from the subprogram level.

SUBROUTINE CALLING ITSELF *BASIC*

A CALL statement has been found which references the subprogram in which it resides. Recursive calls in any form are prohibited.

SUBROUTINE LIMIT OF 30 EXCEEDED *BASIC*

BASIC will not accept more than 30 subprograms. Combine several subprograms or change program logic to eliminate a few.

SYSTEM CLOSED TO NEW USERS, TRY LATER *BEM*

The computer operator has closed the system so that no new users will be allowed on. Wait until later to LOGON.

SYSTEM COMMAND NOT RECOGNIZED *BEM*

A command was entered in monitor mode which was not recognized. All commands must begin with one slash and only commands listed below are allowable:

/DELETE *file-info*
/DISPLAY JOBS
/DISPLAY VOLUMES
/EXEC *program*
/FSTATUS *file-info*
/HELP
/INTR
/LOGOFF
/PAUSE *comment*
/PRINT *file-info*
/PUNCH *file-info*
/RUN *program*
/RESUME
/SCREEN
/STATUS RESOURCE
/STATUS TERM
/TYPE *comment*
/VTOC *volume*

"TAB" CANNOT BE USED WITH "PRINT USING" *BASIC*

The TAB function cannot be used while PRINT USING is active. The TAB should be removed, or a semicolon placed before the function call to terminate the USING clause.

TANGENT/COTANGENT OUT OF RANGE *BASIC*

The result of a TAN or COT function evaluation caused an overflow condition. Machine infinity is supplied and execution continues.

TASK ENTERED IN BACKGROUND MODE *BEM*

This is a confirmational message indicating that the Enter file was successfully queued for execution. The task may already have begun, or may be delayed until a batch processor becomes available.

TERMINAL ALREADY LOGGED ON, PROCEED *BEM*

The previous user did not LOGOFF; this terminal is still logged on.

TERMINAL IDLE TOO LONG, REPLY OR BE CANCELLED *BEM*

This terminal has had no activity for a long period of time, and is assumed to have been left idle. If this terminal is still in use, reply within 30 seconds, or BEM will log the terminal off. The time limit before this message is displayed is set by the system administrator.

THE SUBROUTINE DEFINED IS NOT REFERENCED *BASIC*

This is an informational message only. It notifies the user that he has included a subprogram (either implicitly via LIBRARY or explicitly in the workspace) which is never called. It should be eliminated as it only takes up memory. Compilation continues.

TIME UP — PROGRAM LOOPING *BASIC*

The time limit specified in the TIME statement has been exceeded by the program. It may be looping, or it may require that the time limit be increased.

TOO MANY TAB STOPS *EDT*

More than eight tab stops have been used with the TAB command. See Section 3.2.3 in UA-0141.

TYPE OF FUNCTION PARAMS INCONSISTENT IN CALL *BASIC*

The functions passed to a subprogram do not agree in type or number of parameters expected. Check the CALL statements to see that any functions passed contain the same number and type of parameters, then check the subprogram to be sure it references the function correctly.

UNABLE TO CREATE SPOOL FILE *RSP*

RSP could not successfully build the desired Spool file. Check parameters and retry. If the problem persists, consult your Sperry Univac representative.

UNCORRECTED ERROR IN SOURCE *BASIC*

One of the statements flagged during the previous OLD command has not been eliminated or corrected. The number of that line is shown.

UNKNOWN ERROR ON SAT FILE *LIBRARY*

BEM has received an error code from the SAT processor which it does not expect. If the command issued does not violate any of the constraints placed on it by BEM, contact your local Sperry Univac representative.

UNRECOGNIZABLE COMMAND *EDT*

A command keyword has been used which the Editor does not recognize. Check all keywords used against Appendix A of UA-0141.

USER DID NOT SUPPLY FILE NAME & NO DEFAULT GIVEN *FILE/LIBRARY*

The user has issued a command which requires that a file-name be specified; no filename was stated in the command. If the user expected to use a default file specification, he should contact the system administrator, as the administrator did not declare a default file for this account. To correct the command, enter a filename explicitly.

USER LOGGED OFF, CANCELLED BY OPERATOR *BEM*

The operator has cancelled your task for some reason. Contact the operator to find out why.

USER LOGGED OFF, END OF FILE ON CONTROL STREAM *BEM*

This message is only issued by a batch processor. It indicates that the processor attempted to read more cards from the enter stream and found none left. The enter task is automatically logged off. This is usually caused by a misinterpreted or mistyped LOGOFF command.

USER LOGGED OFF, NO RESPONSE IS ALLOTTED TIME *BEM*

This message is issued 30 seconds after the "TERMINAL IDLE TOO LONG. . ." message if no response is made. To use this terminal, the next user need only log on again.

USER LOGGED OFF, TERMINAL IS NOW FREE *BEM*

This indicates successful completion of a LOGOFF command.

USER'S ACCOUNT DOES NOT PERMIT WRITING TO FILES *FILE/LIBRARY*

The user has attempted to write or update a file and is not permitted to by his account description. To remove this restriction, contact the system administrator to change the access permission.

USER'S ACCOUNT PROHIBITS ACCESS TO THIS FILE *FILE/LIBRARY*

The account description for this user does not permit the specified file to be accessed. This usually is a result of accessing a file other than the default file if only that file is permitted. To remove this restriction, contact the system administrator.

VOLUME IS NOT AVAILABLE TO THE BEM SYSTEM *LIBRARY/FILE*

The disk volume you have requested is mounted, but has not been made available to the BEM system by the system administrator. Contact the system administrator to have the pack included.

VOLUME NAME IS NOT SPECIFIED *LIBRARY/FILE*

The user has not supplied the name of the disk volume to scan, and the volume name is not in the catalog.

VOLUME NOT MOUNTED ON A DISK DRIVE *LIBRARY/FILE*

The volume requested is not mounted on a disk drive. Contact the operator to have the volume mounted.

WAITING FOR OPEN FILE TABLE ENTRY *LIBRARY/FILE*

An ''Open File Table Entry'' in the preamble could not be secured for this file access. BEM will wait until another file access terminates and releases its entry. If the user does not wish to wait, the interrupt facility of BEM may be used to terminate the file access. If this problem occurs frequently, contact the system administrator to have more entries placed in the preamble.

# OF FUNCTION PARAMS INCONSISTENT IN CALL *BASIC*

The number of parameters passed to a subprogram does not agree with the number stated on the SUB line, or does not agree with another CALL to the same program.

#0 INVALID ON CHAIN *BASIC*

Channel zero, the terminal, may not be used as the file from which the chained program is to be read. A data management, temporary, or library file must be used.

APPENDIX D SAMPLE EDT SESSIONS

Examples of complete sessions are provided in this appendix to aid the new user in learning EDT. The designation IN: denotes text which is supplied by the user, and OUT: to designate responses from the system.

Session 1

```
IN:    /LOGON USER
OUT:   USER LOGGED IN, SYSTEM READY
OUT:   /
IN:     EXEC EDT
OUT:   OS/3 EDITOR READY (VER 1.1) BEGIN:
OUT:    1.0000
IN:    @SET TABS=7,CHAR=;
OUT:    1.0000
IN:    C--    SAMPLE FORTRAN PROGRAM TO READ AND PRINT
OUT:    2.0000
IN:    ;DIMENSION A(80)
OUT:    3.0000
IN:    10;READ (5,100,END=30) (A(I),I=1,80)
OUT:    4.0000
IN:    ;WRITE (6,200) (A(I),I=1,80)
OUT:    5.0000
IN:    ;GOTO 10
OUT:    6.0000
IN:    100;FORRAT(80A1)
OUT:    7.0000
IN:    30;STOP
OUT:    8.0000
IN:    ;END
OUT:    9.0000
IN:    @PRINT
OUT:    1.0000 C--    SAMPLE FORTRAN PROGRAM TO READ AND PRINT
OUT:    2.0000        DIMENSION A(80)
OUT:    3.0000 10     READ (5,100,END=30) (A(I),I=1,80)
OUT:    4.0000        WRITE (6,200) (A(I),I=1,80 )
OUT:    5.0000        GOTO 10
OUT:    6.0000 100    FORRAT (80A1)
OUT:    7.0000 30     STOP
OUT:    8.0000        END
OUT:    9.0000
IN:    @ON 6 CHANGE 'FORRAT' TO 'FORMAT'
OUT:    9.0000
IN:    @ 6.1
OUT:    6.1000
IN:    200;FORMAT (' ',80A1)
OUT:    6.2000
IN:    @PRINT 6-7
OUT:    6.0000 100    FORMAT (80A1)
OUT:    6.1000 200    FORMAT (' ',80A1)
OUT:    7.0000 30     STOP
OUT:    6.2000
IN:    @WRITE FORTPROG,SOURCELIB,SYS001,S
OUT:    6.2000
IN:    @HALT
OUT:   /
IN:     LOGOFF
OUT:   USER LOGGED OFF, TERMINAL IS NOW FREE
```

Explanation:

Lines 1-3	These lines constitute the log on procedure. A user has logged on with a user-id of USER.
Lines 4-6	The Editor is invoked.
Line 7	The TABS are set; only column 7 is specified and the character is a semicolon.
Line 8	The Editor responds with a line number of 1.0000 since no text has been entered.
Line 9	The first line of the user's program is entered.
Line 10	Line 2.0000 indicates one line has been stored, and the current line number has been advanced by 1.0 from 1.0000 to 2.0000.
Lines 11-24	The remainder of the FORTRAN program is entered one line at a time. Each time the Editor increments the current line number to reflect the newest free line. Note the use of the semicolon to advance to column 7.
Lines 25-34	A PRINT command is used to display the program. Each line is listed along with the number of the line it was stored at.
Lines 35-36	To correct a spelling error, the CHANGE command is used. FORRAT is corrected to FORMAT.
Lines 37-38	The user wishes to insert a line between lines 6 and 7, so the current line number is reset to 6.1. Since one digit appears to the right of the decimal point, the line increment is also reset to 0.1.
Lines 39-40	A line is inserted at 6.1. The current line number is advanced by 0.1 to 6.2.
Lines 41-45	To verify that the changes are correct, lines 6 to 7 are printed.
Lines 46-47	The new program is written to a Disk file. The program name used is FORTPROG, and it will be added to a library named SOURCELIB on SYS001.
Lines 48-51	The Editor is halted and the user logs off.

Session 2

```
 1 IN:    /LOGON U100
 2 OUT:   USER LOGGED ON, SYSTEM READY
 3 OUT:   /
 4 IN:     EXEC EDT
 5 OUT:   OS/3 EDITOR READY (VER 1.1) BEGIN:
 6 OUT:      1.0000
 7 IN:    @READ ASMPRG1,ASMLIB,DISK02
 8 OUT:   2476.0000
 9 IN:    @COLUMN 1-9 PRINT 'OPENFILE'
10 OUT:   1435.0000 OPENFILE OPEN  PRNTR
11 OUT:   2476.0000
12 IN:    @ON 1435 COL 40 INSERT 'OPEN PRINTER FILE'
13 OUT:   2476.0000
14 IN:    @1435.4
15 OUT:   1435.4000
16 IN:    @SET TABS=10,16,40,CHAR=;
17 OUT:   1435.4000
18 IN:    ;OPEN;READER;OPEN CARD FILE
19 OUT:   1435.5000
20 IN:    @WRITE ASMPRG1,LIBRARY,DISK02
21 OUT:   OVERWRITE (YES OR NO)?
22 IN:    YES
23 OUT:   1435.5000
24 IN:    @DELETE
25 OUT:      1.0000
26 IN:    @READ ASMJCL,JCLIB,SYS005
27 OUT:     12.0000
28 IN:    @PRINT
29 OUT:      1.0000 // JOB ASSEMBLA,,10000
30 OUT:      2.0000 // DVC 20 // LFD PRNTR
31 OUT:      3.0000 // DVC 50 // VO DISK02 // LBL LIBRARY
32 OUT:      4.0000 // LFD ASMLIB
33 OUT:      5.0000 // WORK1
34 OUT:      6.0000 // WORK2
35 OUT:      7.0000 // EXEC ASM
36 OUT:      8.0000 // PARAM IN=ASMPRG1/ASMLIB
37 OUT:      9.0000 // PARAM OUT=ASMLIB
38 OUT:     10.0000 /&
39 OUT:     11.0000 // FIN
40 OUT:     12.0000
41 IN:    @COPY 1-11 TO 12 CHANGE 'ASSEMBLA' TO 'ASSEMBLB'
42 OUT:     23.0000
43 IN:    @ON 12-22 CHANGE 'ASMPRG1' TO 'ASMPRG2'
44 OUT:     23.0000
45 IN:    @COPY 1-11 TO 23 CHANGE 'ASSEMBLA' TO 'ASSEMBLC'
46 OUT:     34.0000
47 IN:    @ON 23-33 CHANGE 'ASMPRG1' TO 'ASMPRG3'
48 OUT:     34.0000
49 IN:    @COPY 1-11 TO 34 CHANGE 'ASSEMBLA' TO 'ASSEMBLD'
50 OUT:     45.0000
51 IN:    @ON 34-44 CHANGE 'ASMPRG1' TO 'ASMPRG4'
52 OUT:     45.0000
53 IN:    @WRITE 1-11 JCL1,JCLIB,SYS005
54 OUT:     45.0000
55 IN:    @WRITE 12-22 JCL2,JCLIB,SYS005
56 OUT:     45.0000
57 IN:    @WRITE 23-33 JCL3,JCLIB,SYS005
58 OUT:     45.0000
```

```
59 IN:    @WRITE 34-44 JCL4,JCLIB,SYS005
60 OUT:     45.0000
61 IN:    @HALT
62 OUT:   /
63 IN:     LOGOFF
64 OUT:   USER LOGGED OFF, TERMINAL IS NOW FREE
```

Explanation:

Lines 1-6	The user has logged on using a user-id of U100, and invokes the Editor.
Lines 7-8	An element named ASMPGM1 is read into the Work file to be edited from a library called ASMLIB on the disk DISK02. The Work file has 2475 lines copied into it. Thus, the current line number is reset to 2476.
Lines 9-11	To locate the definition of the label OPENFILE the PRINT command is used. In assembly language all definitions occur in columns 1 to 9 of the text, so the user has restricted the search to these columns. In this manner, references to the label OPENFILE will not be displayed since they do not occur in columns 1 to 9. A single definition is found at line 1435, and it is displayed.
Lines 12-13	Comments in assembly language generally begin in column 40. The command listed inserts the comment OPEN PRINTER FILE at line 1435.
Lines 14-19	In order to insert a statement between lines 1435 and 1436 the current line number is reset. The user has chosen to set it to 1435.4, however any valid line number between 1435 and 1436 may be used. Tabs are set to allow the user to enter the statement without typing in spaces which would be required to maintain the listing format.
Lines 20-23	The updated element is now written back to the library. The YES response indicates that the user wishes to overwrite the previous version with the new one.
Lines 24-27	Another module is to be updated so the DELETE command is used to erase the old text (it is no longer needed since it has been written to disk). A model assembly job control stream is read in to be duplicated.
Lines 28-40	Before any changes are made the control stream is displayed on the UNISCOPE screen.
Lines 41-42	The JCL is duplicated. A new copy is created beginning at line 12. As it is copied the job name is changed from ASSEMBLA to ASSEMBLB.

Lines 43-44	A change is also made to the name of the program to be assembled. It is changed from ASMPRG1 to ASMPRG2.
Lines 45-52	Two additional copies are made called ASSEMBLC and ASSEMBLD, which assemble programs named ASMPGM3 and ASMPGM4.
Lines 53-60	Each of the new job control streams is written to the disk with the element names JCL1, JCL2, JCL3, and JCL4.
Lines 61-64	The Editor is terminated and the session ended.

The previous example illustrates how job control streams may be stored on disk, and modified as needed to suit a particular application. The control streams may be added to Library files as source elements by filing the JCL to the YJCS library (by using the FILE console command) and copying the control stream using the librarian. The control streams may also be entered via the Editor and written directly to a Library file. To copy the JCL streams from a Library file to the JCS library, the following control stream may be employed.

```
// JOB   COPYJCL
// DVC   50  //   VOL PACK01
// LBL   SOURCELIB   // LFD   LIB
// DVC   20  //   LFD PRNTR
// EXEC   LIBS
/$
    FIL   D0=LIB, D1=$Y$JCS
    COP   DO, S, JCL1, D1
/*
/&
// FIN
```

This will copy the control stream JCL1 in the file SOURCELIB on PACK01 to the JCS library, where it may be run.

Session 3

```
 1 IN:    /LOGON TSTA
 2 OUT:   USER LOGGED ON, SYSTEM READY
 3 OUT:   /
 4 IN:     EXEC EDT
 5 OUT:   OS/3 EDITOR READY (VER 1.1) BEGIN:
 6 OUT:    1.0000
 7 IN:         THIS IS AN EXAMPEL OF HOW TEXT MAY
 8 OUT:    2.0000
 9 IN:    ENTERED AND STORED ASA SOURCE ELEMENT.
10 OUT:    3.0000
11 IN:    IN THIS EXAMPEL SEVERAL MISTAKES HAVE
12 OUT:    4.0000
13 IN:    BEEN MADE IN ENTERING THE PARAGRAPH, BUT
14 OUT:    5.0000
15 IN:    ARE EASILY CORRECTED USING THE EDITOR.
16 OUT:    6.0000
17 IN:    AFTER THE EXAMPEL HAS BEEN COMPLETED, IT
18 OUT:    7.0000
```

```
19 IN:   MAY BE STORRED ON DISC AND PRINTED
20 OUT:   8.0000
21 IN:   USING THE LIBRARIAN. THIS IS USEFUL IN
22 OUT:   9.0000
23 IN:   PREPARING DOCUMENTATION AS IT MAY EASILY
24 OUT:  10.0000
25 IN:   BE ON THE 90/30 PRINTER.
26 OUT:  11.0000
27 IN:   @PRINT
28 OUT:   1.0000      THIS IS AN EXAMPEL OF HOW TEXT MAY
29 OUT:   2.0000 ENTERED AND STORED ASA SOURCE ELEMENT.
30 OUT:   3.0000 IN THIS EXAMPEL SEVERAL MISTAKES HAVE
31 OUT:   4.0000 BEEN MADE IN ENTERING THE PARAGRAPH, BUT
32 OUT:   5.0000 ARE EASILY CORRECTED USING THE EDITOR.
33 OUT:   6.0000 AFTER THE EXAMPEL HAS BEEN COMPLETED, IT
34 OUT:   7.0000 MAY BE STORRED ON DISC AND PRINTED
35 OUT:   8.0000 USING THE LIBRARIAN. THIS IS USEFUL IN
36 OUT:   9.0000 PREPARING DOCUMENTATION AS IT MAY EASILY
37 OUT:  10.0000 BE ON THE 90/30 PRINTER.
38 OUT:  11.0000
39 IN:   @ON & CHANGE ALL 'EXAMPEL' TO 'EXAMPLE'
40 OUT:  11.0000
41 IN:   @ON 1 CHANGE 'MAY' TO 'MAY BE'
42 OUT:  11.0000
43 IN:   @ON 2 CHANGE 'AND' TO 'EDITED AND'
44 OUT:  11.0000
45 IN:   @ON 2 CHANGE 'ASA' TO 'AS A'
46 OUT:  11.0000
47 IN:   @ON 5 INSERT 'THEY' COL 1
48 OUT:  11.0000
49 IN:   @ON 7 CHANGE 'STORRED' T 'STORED'
50 OUT:  11.0000
51 IN:   @ON 10 CHANGE 'BE' TO 'BE REPRODUCED'
52 OUT:  11.0000
53 IN:   @PRINT
54 OUT:   1.0000     THIS IS AN EXAMPLE OF HOW TEXT MAY BE
55 OUT:   2.0000 ENTERED, EDITED AND STORED AS A SOURCE ELEMENT.
56 OUT:   3.0000 IN THIS EXAMPLE SEVERAL MISTAKES HAVE
57 OUT:   4.0000 BEEN MADE IN ENTERING THE PARAGRAPH, BUT
58 OUT:   5.0000 THEY ARE EASILY CORRECTED USING THE EDITOR
59 OUT:   6.0000 AFTER THE EXAMPLE HAS BEEN COMPLETED, IT
60 OUT:   7.0000 MAY BE STORED ON DISC AND PRINTED
61 OUT:   8.0000 USING THE LIBRARIAN. THIS IS USEFUL IN
62 OUT:   9.0000 PREPARING DOCUMENTATION AS IT MAY EASILY
63 OUT:  10.0000 BE REPRODUCED ON THE 90/30 PRINTER.
64 OUT:  11.0000
65 IN:   @WRITE 'DOCUMENT,DOCLIB,PACK01
66 OUT:  11.0000
67 IN:   @HALT
68 OUT:  /
69 IN:    LOGOFF
70 OUT:  USER LOGGED OFF, TERMINAL IS NOW FREE
```

Explanation:

Lines 1-6	The user has logged on with a user-id of TSTA and invokes the Editor.
Lines 7-26	A paragraph of text is entered. As each line is sent to the Editor, the current line number is incremented and displayed to the terminal user. Ten lines are stored, however, several of them contain errors.
Lines 27-38	Before beginning to edit the text, the user requests that the entire paragraph be printed.
Lines 39-40	On several lines, the word "example" was misspelled. Rather than specify each line to be corrected, the user requests that the Editor scan all lines and make the change where necessary. An & is used to designate that all lines are to be scanned.
Lines 41-42	The word "be" is inserted after the word "may".
Lines 43-44	The word "edited" is inserted between "entered" and "and".
Lines 45-46	A space was inserted to change "asa" to the words "as a".
Lines 47-48	The word "they" is inserted at the beginning of line 5.
Lines 49-50	The CHANGE command is used to correct a spelling error.
Lines 51-52	The word "reproduced" is inserted on line 10.
Lines 53-64	A final display is requested to verify that all changes were made correctly.
Lines 65-66	The paragraph is saved on a Library file. It is given a name of DOCUMENT and placed in the Library file DOCLIB on PACK01.
Lines 67-70	The Editor is terminated and the session ended.

Session 4

Example of Procedures

The following is a sample proc which replaces symbolic labels in procs with the absolute line number.

```
 1  IN: @ PROC 1
 2  IN: @@ COL 1-1 FIND '/'
 3  IN: @@IF .F.GOTO 99
 4  IN: @@ASSIGN G1=?:1-2
 5  IN: @@A G2=?
 6  IN: @@CHANGE'#G1'TO''ON#G2
 7  IN: @@ CHANGE ALL'#G1'TO'#G2'
 8  IN: @@GOTO 1
 9  IN: @99:@@NOP
10  IN: @END
```

This proc will allow proc-writers to use symbolic names consisting of a slash followed by any character, in place of absolute line numbers. To do so type a proc using / labels, then DO the above proc to change the / labels to line numbers. A / label consists of a slash (/) followed by any single character. After the label must be a valid EDT command line.

Explanation:

Line 1	Puts the Editor into Proc file 1.
Lines 2-8	Entry of proc commands as data lines.
Line 9	Definition of line 99 as the last line in the Proc file.
Line 10	Return Editor to proc zero.

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Application Services for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

____
(Document Title)

(UA No.)	*(Revision No.)*	*(Update No.)*

Comments:

From:

____
(Name of User)

____
(Business Address)

Fold on dotted lines, staple, and mail. (No postage stamp necessary if mailed in U. S. A.)
Thank you for your cooperation.

Cut along line.

Staple

Staple

Fold

FIRST CLASS
Permit No. 21
Blue Bell, Pa.

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

Postage will be paid by

SPERRY UNIVAC

Attn: Manager, Application Services

P. O. Box 500

Blue Bell, PA 19424

Fold

Cut along line.