Mark I
Time-Sharing Service

Reference Manual

Information Systems

Time-Sharing Service

Editing Commands

GENERAL ELECTRIC

INFORMATION SERVICE DEPARTMENT

708209B

Editing Commands

August 1967
Revised 1-68, 7-68

INFORMATION SYSTEMS

GENERAL ELECTRIC

Preface

The editing commands described in this manual are an integral part of the Time-Sharing Service of General Electric's Information Service Department. The manual thoroughly describes each editing command and provides examples of how the commands work. This revision describes the new command EDIT TEXT, the new control word. LINK, and the increased flexibility of the DELETE and EXTRACT functions.

This manual is written for present users of the Service, and familiarity with the Command System: Reference Manual (229116A) is assumed.

If this reference manual is used as a self-teaching aid, it may be desirable to consider the editing functions in the following order:

Line Functions

EDIT LIST
EDIT PAGE
EDIT TEXT
EDIT DELETE
EDIT RESEQUENCE
EDIT EXTRACT
EDIT MOVE
EDIT DUPLICATE
EDIT MERGE
EDIT WEAVE
EDIT RUNOFF

String Functions

$FIND
$INSERT
$END
$REPLACE
$LIST
$RUNOFF
$MOVE
$DUPLICATE
$LOCATE
$BEGIN
$BREAK
$IGNORE
$TIME
$TEXT
$PROGRAM
$STRING
$SUBSTITUTE
$MULTIPLE
$ABORT
$TRANSLATION

(20M) 7-68

Contents

Introduction

The editing commands described in this manual permit you to retrieve, modify, and manipulate information stored in the time-sharing system.

These commands are an expansion of editing functions that were previously available with time-sharing. Even though you are an experienced user of time-sharing and are familiar with some of the commands, you should read the fuller descriptions of their usefulness and application contained in these pages.

The commands operate on files of information and are not concerned with the subject matter of the files. It makes very little difference, therefore, whether you are working in BASIC, FORTRAN, ALGOL or in none of these. When you resequence line numbers while using BASIC, all references to line numbers in your file will be changed to reflect the new sequence. This is the case only with BASIC and is not true of FORTRAN or ALGOL. Except for this, the effect of the commands is the same in any of the systems or in an English text. Since the commands operate on files of information and these files are normally programs and have line numbers for reference, line numbers are used to control certain editing processes. Other processes, however, do not depend on line numbers but operate on designated strings of information so that purely textual information can be edited.

Bear in mind that many of the commands modify working copies of time-sharing programs. You should take the precaution of saving important programs before editing them. Similarly, none of these commands automatically save programs after editing has been completed. You must do this yourself.

If you have occasion to need them, you can obtain immediate hints regarding the use of editing commands while at the teletypewriter. Simply type EDIT and the system will type a summary of the commands with an option to receive more information about any of them.

The format illustrations at the beginning of the description of each command are meant to show the components of the particular commands and should not be considered models for entering the commands on a teletypewriter, as to the use of spaces or particular characters. The examples for each of the commands that follow the descriptions show the proper formats for entering on the teletypewriter.

1. Line Functions

Line functions enable you to edit a program on a line-by-line or group-of-lines (block) basis. In addition to line functions, there are several commands which are concerned with entire programs or a series of programs. These commands are included under LINE FUNCTIONS since they are most frequently used while performing EDIT changes by lines or by groups of lines.

The function formats require a blank between the word 'EDIT' and the command word and also a blank between the command word and the argument list. These functions are listed alphabetically and no order of importance should be given to the grouping of the commands. Eleven functions are described and illustrated under LINE FUNCTIONS.

1. Deletions within a Program,
   EDIT DELETE
   EDIT EXTRACT

2. Duplicating Program Lines,
   EDIT DUPLICATE

3. General Functions,
   EDIT RUNOFF

4. Listing Program Lines,
   EDIT LIST

5. Listing Programs on Numbered Pages,
   EDIT PAGE
   EDIT TEXT

6. Merging Programs Together,
   EDIT MERGE
   EDIT WEAVE

7. Rearranging Program Lines,
   EDIT MOVE

8. Resequencing Line Numbers of a Program,
   EDIT RESEQUENCE

DELETE

EDIT DELETE N1, N2-N3, N4 ...

Single lines and blocks of lines can be removed from a program file. Only the specified lines are deleted; all others are saved. To remove a series of successive lines, specify the inclusive beginning and ending line numbers separated by a hyphen. The parameters (N1, N2-N3, and N4 ...) must be separated by commas in the command instruction.

Line numbers, indicating the beginning and ending of a series of line numbers, should be listed in sequential order. However, single line numbers need not be listed sequentially.

CAUTION: An EDIT DELETE command permanently changes your current program. For a new program, it is recommended that you save it before issuing an EDIT DELETE command. To save the edited version of an old program, rename the program and save it. This enables you to retain the old version of this program as well as the new one.

Example 1: Deleting Single Lines

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END


EDIT DELETE 60,30,70,90

READY.

LISTNH


10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Example 2: Deleting Consecutive Lines and Single Lines

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT DELETE 30,60-70,90

READY.

LISTNH


10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

DUPLICATE

EDIT DUPLICATE N1-N2, N3, N4 ...
or N1,N2, N3, N4 ...

Use EDIT DUPLICATE to repeat a single line or a series of lines in an existing program. With this command, all duplicated lines are retained in their original positions as well as duplicated in the required positions. Another command, EDIT MOVE, causes the designated line or block of lines to be removed from their original location and inserted at the new location.

The block of lines between N1 and N2 inclusive are inserted in the original program after the lines specified N3, N4 ... As many insert lines can be utilized as can be specified on the input command line.

If you wish to duplicate a single line, you need only give the line number to be duplicated as the first parameter. In the command EDIT DUPLICATE 10, 25, 40, 50 ..., line 10 will be duplicated following lines 25, 40 and 50. The insert line numbers may be given in any order.

After duplication, the new program is resequenced. If you are editing a BASIC program, the line-number references are changed to reflect the resequenced line numbers. If you are using any operating system other than BASIC, the line-number references of the new program will not reflect the resequenced line numbers.

Example: Duplicating Lines in a BASIC Program

```
10 LET X = 0
20 LET Y = 0
25 REM.....................................................REM
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT DUPLICATE 25,50,80

READY.


LISTNH

100 LET X = 0
110 LET Y = 0
120 REM.....................................................REM
130 REM INITIALIZE X AND Y
140 LET X = X+1
150 LET Y = Y+X
160 REM.....................................................REM
170 REM INCREMENT X BY ØNE
180 REM SUM CØNSECUTIVE INTEGERS
190 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
200 REM.....................................................REM
210 REM SUM FIRST TEN INTEGERS ØNLY
220 IF X = 10 THEN 240
230 GØ TØ 140
240 END
```

EXTRACT

EDIT EXTRACT N1, N2-N3, N4 . . .

You can retain single lines and series of lines in a program file by using EDIT EXTRACT. The lines to be retained are specified; all others are deleted. To retain a series of lines, specify the inclusive beginning and ending line numbers separated by a hyphen.

As many lines, or series of lines, will be retained as specified on a single command line. Only those numbers indicating a series of lines must be in sequential order, all single line numbers have no restriction as to listed order.

CAUTION: An EDIT EXTRACT command permanently changes your current program. If your program is new, save it before issuing an EDIT EXTRACT command. To retain the edited version of an old program, rename the program and save it. This enables you to retain the old version of this program as well as the new one.

Example 1: Extracting Specified Lines

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT EXTRACT 30,60-70,90

READY.

LISTNH


30 REM INITIALIZE X AND Y
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
90 REM SUM FIRST TEN INTEGERS ØNLY
```

Example 2: Extracting Blocks of Lines

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT EXTRACT 0-20,80,40-50,100-99999

READY.

LISTNH


10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

LIST

EDIT LIST
EDIT LIST N1, N2-N3, N4 ...
EDIT LIST N1-N2, N3, N5-N4 ...

This command lists single lines or blocks of lines from a current working program. This list can be printed in either forward or reverse order. A line number may occur more than once as a parameter. Block numbers do not have to be in ascending order. As many lines or blocks can be listed as designated on one input command line.

If parameters are not given following the words EDIT LIST, the entire program will be typed in reverse order starting from the end of the program file. A reverse-order list is also obtained for a block of lines when the block numbers are given as N2-N1 where N2 is greater than N1.

Example 1: Use of EDIT LIST without Line Numbers

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END

EDIT LIST

120 END
110 GØ TØ 40
100 IF X = 10 THEN 120
90 REM SUM FIRST TEN INTEGERS ØNLY
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
70 REM SUM CØNSECUTIVE INTEGERS
60 REM INCREMENT X BY ØNE
50 LET Y = Y+X
40 LET X = X+1
30 REM INITIALIZE X AND Y
20 LET Y = 0
10 LET X = 0
```

Example 2: Use of EDIT LIST with Lines and Blocks of Lines

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS ØNLY
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END

EDIT LIST 0-20,40-50,80,100-200

10 LET X = 0
20 LET Y = 0

40 LET X = X+1
50 LET Y = Y+X

80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y

100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Example 3: EDIT LIST with Blocks of Lines in Reverse Order

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END

EDIT LIST 50-40,20-10



50 LET Y = Y+X
40 LET X = X+1

20 LET Y = 0
10 LET X = 0
```

MERGE

EDIT MERGE MAIN, SUB1, SUB2, SUB3 ...
or MAIN, SUB1, N1, SUB2, N2, SUB3 ...

From two to nine programs can be merged within a main program. This enables you to combine several programs, either saved programs or library programs, into one (up to 6144 characters).

The word MAIN in the command represents the primary program and SUB1, SUB2, etc., represent subprograms to be merged into the primary program. N1, N2, etc., represent line numbers in MAIN after which SUB1, SUB2, etc. are to be inserted. Program names and line numbers must be separated by commas.

The EDIT MERGE command resequences the merged program by increments of 10 starting with 100. All programs are merged in the order specified in the command. Insert line numbers are optional. If they are omitted from the command, the designated programs are sequentially inserted after the last line of the MAIN program.

The current program in working storage is ignored and does not have to be one of those merged. If you want to use the current program in the EDIT MERGE command, you must save it before the command is given.

EDIT MERGE does not affect the saved versions of the programs merged. After you have merged the programs, you should list the new version to insure that the merging was executed correctly. If you want to save the new merged program, rename it before placing it in permanent storage, keeping the original and the merged programs intact.

Example 1: Merging Two Programs

Program A

```
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Program C

```
10 REM THIS BASIC PRØGRAM SUMS
20 REM THE FIRST N CØNSECUTIVE
30 REM INTEGERS, WHERE N IS LESS
40 REM THAN ØR EQUAL TØ TEN.
```

```
EDIT MERGE C,A

READY.

LISTNH
```

```
100 REM THIS BASIC PRØGRAM SUMS
110 REM THE FIRST N CØNSECUTIVE
120 REM INTEGERS, WHERE N IS LESS
130 REM THAN ØR EQUAL TØ TEN.
140 LET X = 0
150 LET Y = 0
160 LET X = X+1
170 LET Y = Y+X
180 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
190 IF X = 10 THEN 210
200 GØ TØ 160
210 END
```

Program A is merged following C. Notice that C does not have an END line. If it had, since MERGE does not delete lines, the merged program would have two END lines resulting in an error message when the program is run. You must delete any superfluous END lines that result from merging programs.

Example 2: Merging Two Programs with a Line Number of Zero

Program A

```
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Program C

```
10 REM THIS BASIC PRØGRAM SUMS
20 REM THE FIRST N CØNSECUTIVE
30 REM INTEGERS, WHERE N IS LESS
40 REM THAN ØR ÉQUAL TØ TEN.
```

```
EDIT MERGE A,C,0

READY.

LISTNH
```

```
100 REM THIS BASIC PRØGRAM SUMS
110 REM THE FIRST N CØNSECUTIVE
120 REM INTEGERS, WHERE N IS LESS
130 REM THAN ØR ÉQUAL TØ TEN.
140 LET X = 0
150 LET Y = 0
160 LET X = X+1
170 LET Y = Y+X
180 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
190 IF X = 10 THEN 210
200 GØ TØ 160
210 END
```

Program C is merged with A following line 0, or at the beginning of the merged program. Omission of the zero line number would have resulted in C being after line 120 of program A.

Example 3: Merging Two Programs without a Line Number

Program A

```
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Program C

```
10 REM THIS BASIC PRØGRAM SUMS
20 REM THE FIRST N CØNSECUTIVE
30 REM INTEGERS, WHERE N IS LESS
40 REM THAN ØR ÉQUAL TØ TEN.
```

```
EDIT MERGE A,C

READY.

LISTNH
```

```
100 LET X = 0
110 LET Y = 0
120 LET X = X+1
130 LET Y = Y+X
140 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
150 IF X = 10 THEN 170
160 GØ TØ 120
170 END
180 REM THIS BASIC PRØGRAM SUMS
190 REM THE FIRST N CØNSECUTIVE
200 REM INTEGERS, WHERE N IS LESS
210 REM THAN ØR ÉQUAL TØ TEN.
```

In the merged program above, the END statement is not the final statement. This violates BASIC programming rules. You must be careful when using MERGE to make sure the END statement appears as the final statement.

Example 4: Merging Two Programs with a Line Number Designation

Program A

```
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Program C

```
10 REM THIS BASIC PRØGRAM SUMS
20 REM THE FIRST N CØNSECUTIVE
30 REM INTEGERS, WHERE N IS LESS
40 REM THAN ØR EQUAL TØ TEN.



EDIT MERGE A,C,110

READY.


LISTNH



100 LET X = 0
110 LET Y = 0
120 LET X = X+1
130 LET Y = Y+X
140 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
150 IF X = 10 THEN 210
160 GØ TØ 120
170 REM THIS BASIC PRØGRAM SUMS
180 REM THE FIRST N CØNSECUTIVE
190 REM INTEGERS, WHERE N IS LESS
200 REM THAN ØR EQUAL TØ TEN.
210 END
```

Program C is inserted following line 110 of program A.

MOVE

EDIT MOVE N1-N2, N3
or N1, N3

You can use EDIT MOVE to move a single line or block of lines from its original position to a new position.

N1 or N1-N2 represent the numbers of the lines to be moved. N3 represents the line number after which the line(s) are to be inserted.

When moving a block of lines, the upper and lower limits of the series, N1-N2, must be such that N3 does not fall between the limits. For example, block lines 5-10 can be moved to any place beyond line 11 or preceding line 5.

Following EDIT MOVE for a block of lines, the lines moved are incremented by one when reinserted into the text. If the moved lines fit numerically between the insert number and the line number following the inserted lines, only the inserted lines are resequenced. Should the inserted block of lines be too long to fit between these numbers, the block and the lines that would otherwise be overlapped are resequenced and the message BLOCK TOO LARGE is given. This message only informs you that a larger portion of the program has been resequenced by ones to allow for the insertion.

Example: Line and Block Moves within a Single Program

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END


EDIT MØVE 30,0

READY.


LISTNH

1 REM INITIALIZE X AND Y
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
 100 IF X = 10 THEN 120
110 GØ TØ 40
120 END

EDIT MØVE 60-70,1

READY.

LISTNH

1 REM INITIALIZE X AND Y
2 REM INCREMENT X BY ØNE
3 REM SUM CØNSECUTIVE INTEGERS
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT MØVE 90,3

READY.


1 REM INITIALIZE X AND Y
2 REM INCREMENT X BY ØNE
3 REM SUM CØNSECUTIVE INTEGERS
4 REM SUM FIRST TEN INTEGERS ØNLY
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

PAGE

EDIT PAGE P1, N, P2, P3, ... P9

Up to nine programs saved under your user number or from a system library can be listed by the EDIT PAGE command. The list is consecutively numbered on 8-1/2 in. by 11 in. pages. Up to 50 lines may appear on one page and ten blank lines separate the programs. Each page is separated by dashed lines to indicate trimming edges when subsequently cut into sheets.

The designations P1, P2.... P9 represent program names and N designates the first page number. If this first page designation is omitted, page 1 is assumed and printed at the top of the first page. When there are fewer than 20 lines left at the bottom of a page, the successive program starts on the next page.

RESEQUENCE

EDIT RESEQUENCE N1, N2, N3
or N1, N2-N3, N4

Often when creating a program you find it necessary to resequence the line numbers. This occurs most frequently after you have inserted several lines into the original number sequence or where you move a block from one part of a program to another.

The two formats shown above for the EDIT RESEQUENCE command are as follows:

N1 represents the first line number in the resequenced portion of your file.
N2 represents the first line number of the portion of the file you are resequencing.
N3 represents the increment between the line numbers of the resequenced portion of the file.
N2-N3 represents an inclusive block of numbers to be resequenced (in which case N4 represents the increment).

Line numbers, or series of numbers, are separated by commas. When an inclusive block (N2-N3) is to be resequenced, the numbers are separated by a hyphen.

When you use EDIT RESEQUENCE without parameters, the system resequences the program starting with line number 100 followed by increments of 10. This is identical to issuing the command EDIT RESEQUENCE 100, 0, 10.

CAUTION: When the current program is a BASIC program and there are line number references within the program, be sure the operating system designated is BASIC. This will change the program line references to agree with the resequenced line numbers.

General Rules for using EDIT RESEQUENCE:

1. All line numbers must be separated by commas.
2. The hyphen can only appear in the second parameter.
3. The command given without line numbers or an increment will resequence the program from 100 in increments of 10.
4. The command given without an increment will resequence in increments of 10.
5. Negative resequences within a block are obtained when N3 is less than N2.
6. Do not choose resequence numbers that will produce either a line number less than zero or larger than 99999.

Example 1: Resequencing without Line Numbers or an Increment

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT RESEQUENCE

READY.

LISTNH

100 LET X = 0
110 LET Y = 0
120 REM INITIALIZE X AND Y
130 LET X = X+1
140 LET Y = Y+X
150 REM INCREMENT X BY ØNE
160 REM SUM CØNSECUTIVE INTEGERS
170 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
180 REM SUM FIRST TEN INTEGERS ØNLY
190 IF X = 10 THEN 210
200 GØ TØ 130
210 END
```

Example 2: Resequencing with Only the Starting Line Number

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END



EDIT RESEQUENCE 200

READY.

LISTNH




200 LET X = 0
210 LET Y = 0
220 REM INITIALIZE X AND Y
230 LET X = X+1
240 LET Y = Y+X
250 REM INCREMENT X BY ØNE
260 REM SUM CØNSECUTIVE INTEGERS
270 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
280 REM SUM FIRST TEN INTEGERS ØNLY
290 IF X = 10 THEN 310
300 GØ TØ 230
310 END
```

Example 3: Resequencing Part of a Program

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT RESEQUENCE 50,40

READY.

LISTNH



10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
50 LET X = X+1
60 LET Y = Y+X
70 REM INCREMENT X BY ØNE
80 REM SUM CØNSECUTIVE INTEGERS
90 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 REM SUM FIRST TEN INTEGERS ØNLY
110 IF X = 10 THEN 130
120 GØ TØ 50
130 END
```

Example 4: Resequencing Part of a Program in Increments of 100

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END



EDIT RESEQUENCE 100,100,100

READY.


LISTNH



10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 300
200 GØ TØ 40
300 END
```

Example 5: Resequencing a Block within a Program

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT RESEQUENCE 20,20-100,5

READY.

LISTNH
```

```
10 LET X = 0
20 LET Y = 0
25 REM INITIALIZE X AND Y
30 LET X = X+1
35 LET Y = Y+X
40 REM INCREMENT X BY ØNE
45 REM SUM CØNSECUTIVE INTEGERS
50 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
55 REM SUM FIRST TEN INTEGERS ØNLY
60 IF X = 10 THEN 120
110 GØ TØ 30
120 END
```

Example 6: Resequencing a Block in Reverse Order

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

```
EDIT RESEQUENCE 80,80-60

READY.

LISTNH



10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
70 REM SUM CØNSECUTIVE INTEGERS
80 REM INCREMENT X BY ØNE
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

RUNOFF

EDIT RUNOFF
EDIT RUNOFF N

A printed copy of your saved program may be obtained by the EDIT RUNOFF command. This is printed in a page format, and paginated with line numbers deleted. The optional line number, represented by N in the command, designates the starting line for the text runoff.

The printout format can be governed by control words inserted in the saved program. Should there be no control words in the program, the printout will be in the following format:

1. The left margin is established by the first print position of the teletypewriter.
2. Each single-spaced line contains 60 characters.
3. There are no line numbers.
4. All pages, except the first, have a one-inch top and bottom margin. The first page has a two-inch top margin.
5. Whenever possible, the right margin is justified.

Control Words:

When typing a program in text, it is possible to control the format of the printout. This control is established by the following control words imbedded in the program text. Each control word must be preceded by a period. An entire line must be devoted to the control word.

Control Word	Definition
.LEFT MARGIN N	Starts the output N spaces to the right of the first print position of the teletypewriter. This is initially preset to zero.
.RIGHT MARGIN N	Places no more than N characters (including the left margin) on a line. Normally, the line will be spaced so that exactly N characters will occur on it. The use of this command corresponds exactly to setting the right margin on a typewriter.

Control Word	Definition
.SPACE N	Sets the number of line-feeds following a carriage return. The command .SPACE 2 means double spacing, etc.
.BREAK	Terminates the line preceding .BREAK. The text following the control word begins on a new line. This is executed automatically for a paragraph, a blank line and certain commands.
.INDENT N	Indents the next line N spaces. The break function is automatically called.
.UNDENT N	Undents the next line N spaces (i.e., pushes it out into the left margin). The designation N must be less than the size of the left margin. If N is not present, it will be taken as equal to the size of the left margin. The break function is automatically called.
.CENTER N	Centers the next N input lines between the left and right margins. If any of the input lines do not fit between the margins, they will generate two output lines with each line centered. When N is not present, it is assumed to be one. The position of the left margin remains as established by the previous .INDENT and .UNDENT commands. This command calls the break function.
.PAGE	Starts the following line at the top of the next page. This control word calls the break function before paging.
.SLEW N	Spaces N lines down the page. If there are fewer than N lines left before the bottom margin, you will continue to the top of the next page.
.IGNORE N	Ignores no more than N leading blanks after the line number on each line. It takes N+1 leading blanks on a line to declare a new paragraph. No non-blank characters are ignored.
.LINK P N	Retrieves the saved file named P and continues the EDIT RUNOFF at the next output line. The optional line number N designates the new starting input line. This control word calls the break function before linking to the new file. All other parameters are preserved and continued over to the new file. Any user saved file or listable library file may be linked by .LINK for EDIT RUNOFF.
.LITERAL	Enables you to start a line of text with a period. In typography this is sometimes called a bullet.

Example:

The first pages of the following example show the control words imbedded in a program of text. These pages are followed by the RUNOFF version of the same program.

```
1 .IGNØRE 3
2   .CEN 1
3  EDIT RUNØFF
4
5    THIS IS A DESCRIPTIØN ØF THE EDIT RUNØFF FUNCTIØN.
6
7 .IGNØRE 2
8  CALLS:  EDIT RUNØFF
9          EDIT RUNØFF N
```

```
10
11  .BREAK
12  THE N SPECIFIES A STARTING LINE NUMBER.  IT IS ØPTIØNAL
13  AND WILL BE DESCRIBED IN ANØTHER PARAGRAPH.
14
15  THE BASIC PURPØSE ØF THE FUNCTIØN IS TØ PRØDUCE FRØM
16  A SAVED FILE A NEAT, PAGINATED CØPY ØF TEXTUAL MATERIAL WITH LINE
17  NUMBERS DELETED.
18
19  YØU CAN CØNTRØL THE FØRMAT ØF THE ØUTPUT THRØUGH THE
20  USE ØF CØNTRØL WØRDS IMBEDDED IN THE TEXT.  IF THERE ARE NØ
21  SUCH CØNTRØL WØRDS, THE ØUTPUT WILL BE IN THE FØLLØWING
22  FØRMAT:  NØ LEFT MARGIN, AND A 60 CHARACTER LINE.  THE TØP
23  MARGIN ØF ALL PAGES EXCEPT THE FIRST WILL BE 1 INCH AND THE
24  BØTTØM MARGIN ØF ALL PAGES WILL BE ØNE INCH.  WHENEVER
25  PØSSIBLE THE RIGHT MARGIN WILL BE JUSTIFIED.  THIS IMPLIES
26  FILLING LINES WITH WØRDS FRØM FØLLØWING LINES AND INSERTING
27  EXTRA SPACES BETWEEN WØRDS.  A PARAGRAPH IS RECØGNIZED
28  BY MØRE THAN THE SPECIFIED NUMBER ØF LEADING
29  BLANKS ØN A LINE (ØNE LEADING BLANK WILL BE IGNØRED.)
30  A PARAGRAPH CAUSES A BREAK IN
31  THE PRØCESS ØF FILLING BETWEEN LINES, AND THE LAST SENTENCE
32  IN THE PRECEDING PARAGRAPH WILL NØT BE RIGHT JUSTIFIED.
33
34  A BLANK LINE WILL PRØDUCE A BLANK LINE IN THE ØUTPUT, AND WILL
35  CAUSE THE SAME BREAK AS A NEW PARAGRAPH.
36  .SLEW 1
37
38  .CEN
39                              CØNTRØL WØRDS
40
41  FØRMAT:  AN ENTIRE LINE IS DEVØTED TØ ØNE WØRD,
42  AND THE FIRST NØN-BLANK CHARACTER ØN THE LINE MUST BE A
43  PERIØD.  THE THREE LETTERS AFTER THE PERIØD ARE ALL THAT ARE
44  LØØKED AT, EXCEPT FØR NUMERICAL PARAMETERS
45
46  (IN ALL THE FØLLØWING FØRMS THE QUØTES ARØUND THE CØNTRØL
47  WØRD SHØULD NØT APPEAR IN THE TEXT. NUMERICAL PARAMETERS
48  BETWEEN PARENTHESES ARE ØPTIØNAL.)
49  .LEF 8
50  .RIG 57
51
52  .UND
53  ".LITERAL"
54  IF YØU SHØULD FØR SØME REASØN HAVE A CASE IN WHICH THE FIRST
55  NØN-BLANK CHARACTER ØN A LINE ACTUALLY IS A PERIØD, THIS MUST
56  BE ØN THE PRECEDING LINE, ØR ELSE THE LINE WILL BE IGNØRED.
57
58  .UND
59  ".RIGHT MARGIN N"
60  .BREAK
61  DØ NØT PUT MØRE THAN N CHARACTERS (INCLUDING THE LEFT MARGIN)
62  ØN A LINE.  NØRMALLY THE LINE WILL BE SPACED ØUT SØ THAT EXACTLY
63  N CHARACTERS WILL ØCCUR ØN IT.  THIS CØMMAND
64  CØRRESPØNDS EXACTLY TØ SETTING THE RIGHT MARGIN ØN A TYPEWRITER.
65
66  .UND
67  ".LEFT MARGIN N"
68  ØUTPUT N SPACES BEFØRE THE FIRST CHARACTER ØF EACH LINE.
69  THIS IS PRESET TØ 0.
70
71  .UND
72  ".LEFT MARGIN N"
73  SETS THE NUMBER ØF LINE-FEEDS AFTER A CARRIAGE RETURN.
74  "SPACE 2" IS DØUBLE SPACING, ETC.
75
76  .UND
77  ".BREAK"
```

```
78  DØ NØT RUN THE WØRDS FRØM THE FØLLØWING LINE INTØ THE PRECEDING
79  LINE.  THIS IS EXECUTED AUTØMATICALLY FØR A PARAGRAPH, A BLANK
80  LINE AND CERTAIN CØMMANDS.
81
82  .UND
83  ".INDENT N"
84  .BREAK
85  INDENT THE NEXT LINE N SPACES.  THE BREAK FUNCTIØN IS
86  AUTØMATICALLY CALLED.
87
88 .IGNØRE 1
89 .UND
90 ".UNDENT (N)"
91 .BREAK
92 UNDENT THE NEXT LINE N SPACES (IE., PUSH IT INTØ THE LEFT
93 MARGIN.) N MUST BE LESS THAN THE SIZE ØF THE LEFT MARGIN.  IF
94 N IS NØT PRESENT IT WILL BE TAKEN AS EQUAL TØ THE SIZE ØF THE
95 LEFT MARGIN.  THE BREAK FUNCTIØN IS AUTØMATICALLY CALLED.
96
97 .UND
98 ".CENTER (N)"
99 .BRE
100 CENTER THE NEXT N INPUT LINES BETWEEN THE LEFT AND RIGHT
101 MARGINS.  IF ANY ØF THE INPUT LINES DØ NØT FIT BETWEEN
102 THE MARGINS THEY WILL GENERATE TWØ ØUTPUT LINES, EACH CENTERED
103 IF N IS NØT PRESENT IT IS ASSUMED TØ BE ØNE.
104 THE PØSITIØN ØF THE LEFT MARGIN IS SET AFTER THE
105 INDENT AND UNDENT CØMMANDS HAVE BEEN EVALUATED.
106 THIS CØMMAND CALLS THE BREAK FUNCTIØN.
107
108 .UND
109 ".PAGE"
110 .BRE
111 SPACE TØ THE TØP ØF THE NEXT PAGE.
112 PAGE CALLS THE BREAK FUNCTIØN BEFØRE PAGING.
113
114 .UND
115 ".SLEW N"
116 .BREAK
117 SPACE N LINES DØWN THE PAGE.  IF THERE ARE FEWER THAN N LINES LEFT
118 BEFØRE THE BØTTØM MARGIN.  THE TEXT WILL CØNTINUE TØ THE TØP
119 ØF THE NEXT PAGE.
120
121 .IGN 5 THIS IS AN EXAMPLE ØF THE USE ØF THE IGNØRE FUNCTIØN
122 .UND
123 ".IGNØRE N"
124 .BREAK
125 IGNØRE ØNLY N LEADING BLANKS AFTER THE LINE NUMBER
126 ØN EACH LINE. IE., IT WILL TAKE N+1 LEADING BLANKS ØN
127 A LINE TØ DECLARE A NEW PARAGRAPH.  NØN-BLANK CHARACTERS
128 ARE NØT IGNØRED.
129
130 .UND
131 ".LINK P N"
132 .BREAK
133 PRINT THE CØNTENTS ØF FILE P AT THIS PØINT IN THE RUNØFF
134 STARTING WITH LINE N ØF THE FILE P.
135
136 .LEF 5
137 THE PARAMETER IN THE CALLING LINE (EDIT RUNØFF N) SPECIFIES
138 THE FIRST LINE WHICH CØNTAINS INFØRMATIØN THAT YØU WANT
139 PRINTED.  RUNØFF PRØCESSES ALL THE PRECEDING MATERIAL
140 EDITING IT AND EXECUTING THE CØNTRØL WØRDS, BUT THE FIRST LINE
141 WHICH IT PRINTS WILL BE THE LINE CØNTAINING THE WØRDS WHICH
142 ØCCUR IN LINE N ØF THE INPUT MATERIAL.  THIS LINE MAY CØNTAIN
143 WØRDS WHICH ØCCURRED ØN THE PREVIØUS LINES ØF INPUT MATERIAL
144 SINCE THE WHØLE LINE WILL BE PRINTED.
145
```

```
SAVE

READY.

EDIT RUNØFF
```

EDIT RUNØFF

THIS IS A DESCRIPTIØN ØF THE EDIT RUNØFF FUNCTIØN.

CALLS: EDIT RUNØFF
EDIT RUNØFF N

THE N SPECIFIES A STARTING LINE NUMBER. IT IS ØPTIØNAL AND WILL BE DESCRIBED IN ANØTHER PARAGRAPH.

THE BASIC PURPØSE ØF THE FUNCTIØN IS TØ PRØDUCE FRØM A SAVED FILE A NEAT, PAGINATED CØPY ØF TEXTUAL MATERIAL WITH LINE NUMBERS DELETED.

YØU CAN CØNTRØL THE FØRMAT ØF THE ØUTPUT THRØUGH THE USE ØF CØNTRØL WØRDS IMBEDDED IN THE TEXT. IF THERE ARE NØ SUCH CØNTRØL WØRDS, THE ØUTPUT WILL BE IN THE FØLLØWING FØRMAT: NØ LEFT MARGIN, AND A 60 CHARACTER LINE. THE TØP MARGIN ØF ALL PAGES EXCEPT THE FIRST WILL BE 1 INCH AND THE BØTTØM MARGIN ØF ALL PAGES WILL BE ØNE INCH. WHENEVER PØSSIBLE THE RIGHT MARGIN WILL BE JUSTIFIED. THIS IMPLIES FILLING LINES WITH WØRDS FRØM FØLLØWING LINES AND INSERTING EXTRA SPACES BETWEEN WØRDS. A PARAGRAPH IS RECØGNIZED BY MØRE THAN THE SPECIFIED NUMBER ØF LEADING BLANKS ØN A LINE (ØNE LEADING BLANK WILL BE IGNØRED.) A PARAGRAPH CAUSES A BREAK IN THE PRØCESS ØF FILLING BETWEEN LINES, AND THE LAST SENTENCE IN THE PRECEDING PARAGRAPH WILL NØT BE RIGHT JUSTIFIED.

A BLANK LINE WILL PRØDUCE A BLANK LINE IN THE ØUTPUT, AND WILL CAUSE THE SAME BREAK AS A NEW PARAGRAPH.

CØNTRØL WØRDS

FØRMAT: AN ENTIRE LINE IS DEVØTED TØ ØNE WØRD, AND THE FIRST NØN-BLANK CHARACTER ØN THE LINE MUST BE A PERIØD. THE THREE LETTERS AFTER THE PERIØD ARE ALL THAT ARE LØØKED AT, EXCEPT FØR NUMERICAL PARAMETERS

(IN ALL THE FØLLØWING FØRMS THE QUØTES ARØUND THE CØNTRØL

WØRD SHØULD NØT APPEAR IN THE TEXT. NUMERICAL PARAMETERS BETWEEN PARENTHESES ARE ØPTIØNAL.)

".LITERAL" IF YØU SHØULD FØR SØME REASØN HAVE A CASE IN WHICH THE FIRST NØN-BLANK CHARACTER ØN A LINE ACTUALLY IS A PERIØD, THIS MUST BE ØN THE PRECEDING LINE, ØR ELSE THE LINE WILL BE IGNØRED.

".RIGHT MARGIN N"
DØ NØT PUT MØRE THAN N CHARACTERS (INCLUDING THE LEFT MARGIN) ØN A LINE. NØRMALLY THE LINE WILL BE SPACED ØUT SØ THAT EXACTLY N CHARACTERS WILL ØCCUR ØN IT. THIS CØMMAND CØRRESPØNDS EXACTLY TØ SETTING THE RIGHT MARGIN ØN A TYPEWRITER.

".LEFT MARGIN N" ØUTPUT N SPACES BEFØRE THE FIRST CHARACTER ØF EACH LINE. THIS IS PRESET TØ 0.

".LEFT MARGIN N" SETS THE NUMBER ØF LINE-FEEDS AFTER A CARRIAGE RETURN. "SPACE 2" IS DØUBLE SPACING, ETC.

".BREAK" DØ NØT RUN THE WØRDS FRØM THE FØLLØWING LINE INTØ THE PRECEDING LINE. THIS IS EXECUTED AUTØMATICALLY FØR A PARAGRAPH, A BLANK LINE AND CERTAIN CØMMANDS.

".INDENT N"
INDENT THE NEXT LINE N SPACES. THE BREAK FUNCTIØN IS AUTØMATICALLY CALLED.

".UNDENT (N)"
UNDENT THE NEXT LINE N SPACES (IE., PUSH IT INTØ THE LEFT MARGIN.) N MUST BE LESS THAN THE SIZE ØF THE LEFT MARGIN. IF N IS NØT PRESENT IT WILL BE TAKEN AS EQUAL TØ THE SIZE ØF THE LEFT MARGIN. THE BREAK FUNCTIØN IS AUTØMATICALLY CALLED.

".CENTER (N)"
CENTER THE NEXT N INPUT LINES BETWEEN THE LEFT AND RIGHT MARGINS. IF ANY ØF THE INPUT LINES DØ NØT FIT BETWEEN THE MARGINS THEY WILL GENERATE TWØ ØUTPUT LINES, EACH CENTERED IF N IS NØT PRESENT IT IS ASSUMED TØ BE ØNE. THE PØSITIØN ØF THE LEFT MARGIN IS SET AFTER THE INDENT AND UNDENT CØMMANDS HAVE BEEN EVALUATED. THIS CØMMAND CALLS THE BREAK FUNCTIØN.

".PAGE"
SPACE TØ THE TØP ØF THE NEXT PAGE. PAGE CALLS THE BREAK FUNCTIØN BEFØRE PAGING.

PARAGRAPH. NØN-BLANK CHARACTERS ARE NØT IGNØRED.

".LINK P N"
PRINT THE CØNTENTS ØF FILE P AT THIS PØINT IN THE RUNØFF STARTING WITH LINE N ØF THE FILE P.

THE PARAMETER IN THE CALLING LINE (EDIT RUNØFF N) SPECIFIES THE FIRST LINE WHICH CØNTAINS INFØRMATIØN THAT YØU WANT PRINTED. RUNØFF PRØCESSES ALL THE PRECEDING MATERIAL EDITING IT AND EXECUTING THE CØNTRØL WØRDS, BUT THE FIRST LINE WHICH IT PRINTS WILL BE THE LINE CØNTAINING THE WØRDS WHICH ØCCUR IN LINE N ØF THE INPUT MATERIAL. THIS LINE MAY CØNTAIN WØRDS WHICH ØCCURRED ØN THE PREVIØUS LINES ØF INPUT MATERIAL SINCE THE WHØLE LINE WILL BE PRINTED.

TEXT

EDIT TEXT P1, N, P2, P3, . . . P9

This command performs the same function as EDIT PAGE on up to nine files saved under a user number or listable from the system library. EDIT TEXT substitutes blanks for line numbers and therefore produces a clean paged copy of thosefiles which are more readable without line numbers.

The designations P1, P2, P3,...P9 represent the program names. The letter N designates the first page number. When N is not specified page 1 is printed at the top of the first page.

".SLEW N"
SPACE N LINES DØWN THE PAGE. IF THERE ARE FEWER THAN N LINES LEFT BEFØRE THE BØTTØM MARGIN. THE TEXT WILL CØNTINUE TØ THE TØP ØF THE NEXT PAGE.

".IGNØRE N"
IGNØRE ØNLY N LEADING BLANKS AFTER THE LINE NUMBER ØN EACH LINE. IE., IT WILL TAKE N+1 LEADING BLANKS ØN A LINE TØ DECLARE A NEW

WEAVE

EDIT WEAVE PROG1, PROG2, PROG3 . . .

You may use EDIT WEAVE to combine from two to nine saved programs. These programs are woven together in the sequence of existing line numbers. All original line numbers are retained. The EDIT WEAVE function operates in a manner similar to EDIT MERGE. In EDIT MERGE, however, specific programs can be combined within a main program and the new program is resequenced.

PROG1, PROG2, etc., designate program names. These program names can be given in any order since they are combined sequentially by line numbers.

CAUTION: When weaving programs, there should be no duplicate line numbers in the entire group. If line numbers are duplicated, only one will be retained. Normally, the line that is retained comes from the last program in the list that contains that line number. However, if insert line numbers are used, the line that is retained could possibly come from one of the other programs. You must also take care to avoid duplicate END statements. See comments under MERGE.

Example 1: Weaving Two Programs

Program A

```
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Program B

```
30 REM INITIALIZE X AND Y
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
90 REM SUM FIRST TEN INTEGERS ØNLY


EDIT WEAVE A,B

READY.


LISTNH

10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Example 2: Weaving Two Programs with a Common Line Number

Program A

```
10 LET X = 0
20 LET Y = 0
40 LET X = X+1
50 LET Y = Y+X
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Program B

```
30 REM INITIALIZE X AND Y
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X;
85 PRINT "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
```

```
EDIT WEAVE A,B

READY.

LISTNH
```

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X;
85 PRINT "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

Line 80 in program B is printed. Line 80 in program A disappears.

2. String Functions

Definition of a String

A string is defined as a group of consecutive characters. There are certain Editing Commands that have been specially designed to operate on information of this type, independent of any line number orientation. These commands do not require the use of file line numbers for reference or control. They deal only with strings of information that you specify, and therefore are said to perform "string functions."

Types of Strings

A closed string is one in which all the characters in the string are explicitly defined (up to the limits of the Editing Command). A closed string cannot begin and end with a comma since commas are used as control characters in certain commands.

An open string is one in which only some of the beginning and ending characters of the string are explicitly defined. An open string is composed of two closed strings separated by a comma. The first closed string defines the beginning characters, and the second closed string defines the ending characters of the text string being specified.

Delimiters

To define where a string begins and ends a character called a "delimiter" is used. The position of this character in the format of the Editing Command identifies it as the delimiter. There are certain restrictions in the choice of a delimiter character; it cannot be a blank, or cannot be numeric, nor can it be a dollar sign.

Argument List

Many of the Editing Commands that perform string functions require that certain information follow the command word itself. This information is called the Argument List and it consists of one or more of the following elements:

- a line number (L)
- a string, including its delimiter characters (S)
- a repetition count (R)

Each of these elements is explained below:

The line number is optional. It consists of one to five numerics. Its meaning depends upon the function of the individual Editing Command with which it is associated. When a line number enters into the function of a command, its role is explained in the definition of the command which is discussed below.

The string has the form /ABCXYZ/ where / is the delimiter character. The string may include numerics, blanks, and dollar signs; anything except the delimiter character. Any Carriage Return character included in the string is not considered to be an integral part of the string, but rather is treated as a control character which allows you to continue the string on the next line. Each character in the string is examined upon input to determine if and how its entry should be allowed to proceed. Unless something to the contrary has been specified by a previous Editing Command (such as $SUBSTITUTE, $MULTIPLE, or $BREAK), the identity of the character is maintained.

The repetition count is optional. If it is not given, it is assumed to be "one." The specific function of the repetition count depends on the command with which it is used and these are discussed below.

The argument list can contain a variety of combinations of line numbers, strings, and repetition counts. Only certain combinations are acceptable, however, and these are shown in the following table.

ACCEPTABLE ARGUMENT LISTS

(Brackets indicate that the command assumes "1" if no repetition count is specified)

Closed Strings:

L1	S1	R1	L2	S2	R2
100	""	[1]			
-	"STRING"	[1]			
-	"STRING"	2			
100	"STRING"	[1]			
100	"STRING"	2			

Open Strings:

100	"STRING1"	2 ,	200	""	[1]
100	"STRING1"	2 ,	-	"STRING2"	[1]
100	"STRING1"	[1]	-	"STRING2"	3
100	"STRING1"	2 ,	200	"STRING2"	[1]
100	"STRING1"	2 ,	200	"STRING2"	3
-	"STRING1"	2 ,	200	""	[1]
-	"STRING1"	[1]	-	"STRING2"	[1]
-	"STRING1"	[1]	-	"STRING2"	3
-	"STRING1"	2 ,	200	"STRING2"	[1]
-	"STRING1"	2 ,	200	"STRING2"	3

String Pointers

The user has two imaginary pointers which he can place at each end of a string in the file by using various Editing Commands. Initially, the pointers are at the beginning of the file. All searches for strings begin at the current position of the beginning string pointer and continue to the end of the file. If the pointers are not at the beginning of the file, there will be no search from the beginning of the file to the place where the pointer had been set. The pointers can be reset to the beginning of the file at any time by using the command $BEGIN which is described below.

String Search

A string search consists of comparing the specified string in the argument list to the text characters in a file to determine if a one-to-one correspondence exists. (The one-to-one relationship can be modified, however, in certain commands such as $IGNORE or $BREAK).

String Function Commands

The twenty Editing Commands performing string functions are grouped below according to the type of editing for which they are used.

1. Character Definition

   $MULTIPLE
   $SUBSTITUTE

2. Major Editing

   $DUPLICATE
   $INSERT
   $MOVE
   $REPLACE

3. Pointer Manipulation

   $BEGIN
   $FIND

4. Printing

   $RUNOFF

5. Scan Control

   $ABORT
   $BREAK
   $IGNORE
   $PROGRAM
   $TEXT

6. Termination

   $END

7. User Aid and Status

   $LIST
   $LOCATE
   $STRING
   $TIME
   $TRANSLATION

Each of the above Editing Commands will now be described in detail. The Commands are arranged in alphabetical order for your easy reference.

$ABORT

EDIT $ABORT S1

This function acts as a watchdog on your input. You can specify a single abort character (in essence, a one-character string) with the $ABORT command. Whenever a subsequent function encounters this abort character as part of its input string, that particular function is discontinued and the rest of the input line is ignored.

In the above format, S1 represents the abort character. This abort character is preset to the back-slash \ located as the upper case L. You can replace this with another character, for example the =, by typing: $ABORT /=/. By following this parameter specification, your function request is aborted whenever you use = in the input.

To restore $ABORT to its normal value type $ABORT//.

Example:

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N
```

```
EDIT $ABØRT /↑/ $FIND /A B ↑C/ $REPLACE /X Y Z / $END

ABØRTING $FI
RETRY?
```

All instructions are aborted after ↑ appears in the argument of $FIND.

$BEGIN

EDIT $BEGIN

The $BEGIN function requires no parameters after the command word. This function resets the string pointers to the beginning of the program. It is used when you wish to perform an operation in a location previous to the current position of the string pointers. When in doubt about the location of the string pointers, use the $BEGIN command.

Although initially set to the beginning of a program, the string pointers may be moved by using $FIND, $REPLACE, $INSERT, $MOVE or $DUPLICATE. The only way you can be sure that a string search starts from the beginning of a file is to issue $BEGIN.

You can issue a $BEGIN before or after any other command. However, the functions that require a string for their application ($STRING, $REPLACE, $INSERT, $MOVE, $DUPLICATE) are not executed if they immediately follow a $BEGIN command.

$BREAK

EDIT $BREAK/ABCDEF .../

The $BREAK function allows you to specify characters to be overlooked in the text when searching for a specific string. The primary difference is in the means of specification. With the $BREAK command you specify a break character, which in an input string, takes the place of several characters in the text search.

In the above format, the letter A is the break character which represents the characters in the text indicated by BCDEF.... You can specify as many break characters as you can designate on one line.

A break character may represent itself or any other character in the $BREAK string of which it is the first character. To return all break characters to their initial values, type: $BREAK//.

Example: $BREAK Command Substituting * for "space" and ,

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N



EDIT $BREAK /* ,/ $FIND /F*A/ $REP /Z/ $FIND /L*G/ $REP /X/ $END

TIME:  0:01

READY.


LISTNH



100 A B C D E Z B C D E F
110 G,H,I,J,K,X,H,I,J,K,L
120 M N N N N N N N N N N N
```

Observe that * was substituted for space and , so that the text string /F A/ was interpreted as equivalent to the input string /F*A/ and the text string /L,G/ was interpreted as /L*G/.

$DUPLICATE

EDIT $DUPLICATE L1 S1 R1 L2 S2 R2 ...

Use $DUPLICATE to duplicate a string (defined by $FIND) in one or more locations within a file. Not only is the string retained in its original position but it is also placed immediately following the strings(S1 , S2 ...) defined by $DUPLICATE. $DUPLICATE must always be preceded by $FIND.

Definition of Parameters:

L1 – Optional line number used to locate S1.
S1 – First string after which the $FIND string is to be placed.
R1 – The $FIND string is to be placed after the R1th occurrence of S1.
L2 – Optional line number used to locate S2.
S2 – Second string after which the $FIND string is to be placed.
R2 – The $FIND string is to be placed after the R2th occurrence of S2.

Example 1: $DUPLICATE with R Set at 2

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N


EDIT $FIND /A/ $DUPLICATE /H/ 2 $END

TIME: 0:01

READY.

LISTNH



100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,HA,I J,K,L
120 M N N N N N N N N N N N
```

Example 2: $DUPLICATE with R Set at 7

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N


EDIT $FIND /A/ $DUPLICATE /N/ 7 $END

TIME:  0:01

READY.


LISTNH



100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N NA N N N N
```

In the first example A was found and duplicated after the second occurrence of H. In the second example A was duplicated after the seventh occurrence of N. Both times A was retained in its original position.

$END

EDIT $END

When you have finished entering string commands, the string mode is terminated by $END. $END is a no-parameter function which incorporates your completed string editing into the working file. To place these revisions into your permanent file, type SAVE.

$END destroys the string pointers and all character definitions in the translation table. Following the execution of this command, the system prints the time in minutes and seconds.

$FIND

EDIT $FIND L1 S1 R1
or L1 S1 R1 , L2 S2 R2

$FIND, without doubt, is the most commonly used string function. It must precede the functions $REPLACE, $INSERT, $MOVE and $DUPLICATE.

There are two string pointers. Upon entering string mode, the two pointers are initially set at the beginning of the file. If you search for one string and find it, the string pointers are then located around this string. When you search for the next string, one pointer stays with the first string. Should the new string not be found, the second pointer returns to the point where the search began (the first string). If the new string is found, the two string pointers are then placed around it.

L1 represents the optional line number of the desired string; S1 the delineated string; and R1 the optional repetition count. Similarly, the designations L2, S2, and R2 used in conjunction with L1, S1, and R1, represent an open string.

A string search using $FIND will begin at the current location of the string pointers. If you are in doubt concerning the location of the string pointers issue $BEGIN. This returns the string pointers to the beginning of the program.

$IGNORE

EDIT $IGNORE S1

$IGNORE is one of two scan-control functions (see $BREAK) which allows you to specify characters to ignore when searching for a particular string. S1 represents a string of from one to thirteen characters which the scan will ignore. The order in which the characters are specified is not relevant.

When an input string is compared with a text string, there must be a one-to-one correspondence between input string characters and text string characters (within the limits of the character definition functions). However, it is possible to allow characters to occur in the text string which are not specified in the input string. These characters are referred to as the ignore characters.

All characters in the string S1 of $IGNORE will be ignored in succeeding string searches. If there are no characters in S1, then nothing will be ignored. Individual characters may not be added to or subtracted from the set of ignore characters because the set is redefined with each $IGNORE. The $IGNORE characters must be defined for each search. They are erased as soon as the $LOCATE or $FIND for which they were defined has terminated.

Example: Use of $IGNORE

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N



EDIT $IGNØRE /,/ $FIND /JKL/ $REPLACE /MNP/ 2 $END

TIME: 0:01

READY.



LIST



HRH1        13:34   LA1 THU 6/29/67

100 A B C D E F A B C D E F
110 G,H,I,MNP,G,H,I,MNP
120 M N N N N N N N N N N N
```

The $IGNORE command allowed the search to find JKL by ignoring the commas. Otherwise the search would have been unsuccessful.

$INSERT

EDIT $INSERT S1 R1 S2 R2 S3 R3 ...

$INSERT allows you to insert one character or several characters after the string specified by $FIND. The text is automatically expanded to accommodate the insert. $FIND must always precede $INSERT.

In the above format, string S1 is inserted after the string specified by $FIND. The symbols R1, R2, R3 represent an optional repetition count which specifies the number of times that the strings S1, S2 and S3 are to be inserted after sequential occurrences of the string specified by $FIND.

Example: Use of $INSERT

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N

EDIT $FIND /,/ $INSERT /↑/ 3 /+/ 2 /:/ $END

TIME: 0:01

READY.

LISTNH

100 A B C D E F A B C D E F
110 G,↑H,↑I,↑J,+K,+L,:G,H,I,J,K,L
120 M N N N N N N N N N N N
```

The search found the first comma, inserted ↑ after the first three occurrences, + after the next two, and : after the next one (R when omitted is always assumed to be one).

$LIST

EDIT $LIST L1 S1

In string mode, the $LIST allows you to list a portion of your file or your entire file to check the accuracy of your editing. The $LIST command prints the text beginning with the first character of the string currently specified by the string pointers. The list ends with the last character of the string specified by S1. Therefore, to list selected portions of your file, use $FIND to set the string pointers where you want to begin listing and use S1 to indicate the end of the list.

If no current string is specified, $LIST begins with the first character of the file. The output ends with the last character of the file if S1 is not specified.

The output is enclosed in quotation marks which are not part of the text. $LIST does not change the position of the current string pointers or the translation table. It may precede or follow any string function.

Example 1: Entire File Listed

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N

EDIT $LIST

"100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N"
```

Example 2: Using $FIND to List Part of a File

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N
```

```
EDIT $FIND /L/ $LIST

"L,G,H,I,J,K,L
120 M N N N N N N N N N N N
```

$FIND sets the string pointers at L and the remainder of the file is listed.

$LOCATE

EDIT $LOCATE L1 S1 R1
or L1 S1 R1 , L2 S2 R2

Use $LOCATE to search your file from beginning to end for the specified string S1. This will print a double-spaced list of all lines containing this string of characters. If the optional line number L1 is given, the search begins at that line number and continues through the rest of the file. The designation R1 represents an optional repetition count which directs the printing to begin with the R1'th occurrence of string S1, and to print only lines containing each R1'th occurrence of S1.

The use of $LOCATE does not affect the location of the string pointers. It can precede or follow any other string function.

Example 1: Use of $LOCATE to Print All Occurrences of a Designated String

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END
```

$LOCATE prints all occurrences of the string X.

```
EDIT $LØCATE /X/


LØCATING:
"X" 1


10  LET X = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
100 IF X = 10 THEN 120
```

Example 2: Use of $LOCATE with Repetition Count

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END

EDIT $LØCATE /X/ 2
WAIT.

LØCATING:
"X" 2

30 REM INITIALIZE X AND Y
40 LET X = X+1
60 REM INCREMENT X BY ØNE
100 IF X = 10 THEN 120
```

$LOCATE prints all lines containing every second occurrence of the string X beginning with the second occurrence.

$MOVE

EDIT $MOVE L1 S1 R1

Use the $MOVE command to move a single string from one location to another in a file. Specified by $FIND, this string is deleted from its original position and placed immediately following the string defined by $MOVE. $MOVE must always be preceded by $FIND. The string specified by $FIND is inserted after the R1th occurrence of the string S1. Optional parameter L1 represents the line number associated with string S1.

Example: Use of $MOVE

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N


EDIT $FIND / M/  $MØVE /N/ 5 $END

TIME:  0:02

READY.


LIST



HRH1

100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 N N N N N M N N N N N N
```

Observe that the string following $FIND consists of two characters: space and M. This string is inserted after the fifth occurrence of string N.

$MULTIPLE

EDIT $MULTIPLE S1 S2 S3 ...

$MULTIPLE is one of three character definition functions (refer to $SUBSTITUTE and $BREAK). $MULTIPLE allows you to define one character to have the value of two other characters. If character A is the multiple of character B and C, then all strings with B or C in a corresponding position to A in the input string will match the input string, and A will lose its identity as A. If a two-character string is used with $MULTIPLE, the first character does not lose its identity. Thus,

EDIT $MULTIPLE /XY/ $FIND /XY/

Found	Not Found
XY	AY (unless A has been defined as equal to X or Y)
YY	

EDIT $MULTIPLE /XYZ/ $FIND /XYZ/

Found	Not Found
YYZ	XYZ (X does not keep its identity as X)
ZYZ	AYZ (unless A has been defined as equal to Y or Z)

If X is a multiple of Y and Z, then Y and Z may have their initial values. In addition, either Y or Z or both may be substitute characters. Where Y or Z is a $BREAK character, it is assumed by $MULTIPLE that Y or Z only have their regular input values.

When a $MULTIPLE string has more than three characters, the first character is the multiple of the last two. If the $MULTIPLE string is one character in length, that character is restored to its initial value.

Initially, there are no multiple characters. When $MULTIPLE is issued, the translation changes which have been defined remain in effect until (1) a vacant $MULTIPLE is issued, or (2) the values of the multiple characters are redefined by another $MULTIPLE, $SUBSTITUTE, or $BREAK.

Unlike other character definition functions, strings defined by one set of $MULTIPLE characters may be changed by issuing character definition functions which modify the characters to which the $MULTIPLE characters refer. For example, if X is a multiple of Y and Z, and X is entered in an input string, then the input string will not be affected by redefinition of X, but redefinitions of Y and Z will change the value of the input string.

The series of commands:

EDIT $SUBSTITUTE /X/ $MULTIPLE /YX/ $SUBSTITUTE $FIND /XY/

- defines X to be a carriage return ($SUBSTITUTE /X/)
- defines Y to be Y and a carriage return ($MULTIPLE /XY/)
- defines X to be X and also defines Y to be Y and X ($SUBSTITUTE)
- finds the string XY where Y is Y or X ($FIND /XY/)

A $MULTIPLE with or without parameters may precede or follow any other function.

Examples:

1. $MULTIPLE /XYZ/ — X matches Y and Z
2. $MULTIPLE /XY/ /YZ/ — X matches X and Y
   so X matches X, Y, and Z
   Y matches Y and Z
   Z matches Z
   Note that Z may be a substituted character.
3. $MULTIPLE /X/ — X is restored to its initial value.
4. $MULTIPLE — All multiple characters are restored to their initial values.

Example: Use of $MULTIPLE

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N


EDIT $MULTIPLE  /*,/ $FIND /*/ $REPLACE /---/ 11 $END

TIME:  0:01

READY.



LISTNH


100 A B C D E F A B C D E F
110 G---H---I---J---K---L---G---H---I---J---K---L
120 M N N N N N N N N N N N
```

Observe that * became the multiple of , after which $FIND and $REPLACE replaced eleven occurrences of , with ---.

$PROGRAM

EDIT $PROGRAM

The $PROGRAM command, always expressed without parameters, establishes one of two scan-control modes. This mode, known as the program mode, allows you to search through a file as a continuous string of characters. No characters are ignored when you are locating a string, not even carriage returns or line numbers. This is the normal mode of operation when utilizing string functions. The program mode is automatically established whenever you begin using string functions.

You remain in the program mode of operation until you initiate text mode which is the second scan-control mode (refer to EDIT $TEXT).

$REPLACE

EDIT $REPLACE S1 R1 S2 R2 S3 R3 ...
or $REPLACE S1 *

The $REPLACE command replaces a text string with an input string or it replaces several text strings by one or more input strings. The text strings are identical, and may be dispersed throughout your file. The file expands or contracts automatically to accommodate the replacement strings.

This command replaces the current string, specified by the string pointers, with the input string. After the replacement has occurred, the string pointers are placed around the next occurrence of the text string. If there are no further occurrences of the text string, the pointers are returned to the beginning of the file.

For more than one text string, the replacements are executed in the following manner: the first R1 occurrences of the text string are replaced by the input string S1, the next R2 occurrences of the text string are replaced by the input string S2, etc. If the repetition counts (R1 R2 R3...) are not given, they are assumed to be one.

In the format, EDIT $REPLACE S1 *, the * indicates that the specified text string will be replaced by the string S1 at every location where it occurs in the file, from the current location of the string pointers to the end of the file.

$REPLACE must always be preceded by $FIND. If there is some doubt concerning the location of the string pointers, issue $BEGIN before you issue the $FIND command.

Example 1: Replacement of Several Strings with Three Input Strings.

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N

EDIT $FIND /,/ $REPLACE /*/ 5 /↑/ 2 /:/ 3 $END

TIME:  0:01

READY.

LISTNH

100 A B C D E F A B C D E F
110 G*H*I*J*K*L↑G↑H:I:J:K,L
120 M N N N N N N N N N N N
```

The string /,/ is replaced in five occurrences with /*/, in two occurrences with /↑/, and in three with /:/.

Example 2: Replacement of a String in All of Its Occurrences.

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N

EDIT $FIND /,/ $REPLACE /↑/ * $END

TIME:  0:01

READY.

LISTNH

100 A B C D E F A B C D E F
110 G↑H↑I↑J↑K↑L↑G↑H↑I↑J↑K↑L
120 M N N N N N N N N N N N
```

The string /,/ is replaced with /↑/ everytime it occurs by adding * to the $REPLACE command.

$RUNOFF

EDIT $RUNOFF L1

While operating in string mode, you can obtain a printed copy of your file by using $RUNOFF. It is exactly the same as EDIT RUNOFF except that it is called from the string mode. In the $RUNOFF command, L1 is optional and when specified represents the line number where you wish to begin your runoff.

For an example of EDIT $RUNOFF, refer to EDIT RUNOFF in the preceding section of line functions.

$STRING

EDIT $STRING

$STRING is a no-parameter function which allows you to find where the string pointers are currently located by listing the current string. This listed string is part of the file (the text string) and not merely a copy of the input string.

When the current string is listed, the printed copy is enclosed in quotation marks. These quotation marks are not part of the text string.

The $STRING does not modify the string pointers or the translation table. It may precede o. follow any other function.

Example: Use of $STRING

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N


EDIT $FIND /H,I,/ $STRING


CURRENT STRING:

"H,I,"
```

Observe that $STRING prints the string delineated by the current position of the string pointers.

$SUBSTITUTE

EDIT $SUBSTITUTE S1

$SUBSTITUTE allows you to assign the value of a printable character to the carriage return. The substituted character may be entered into an input string in place of the carriage return. It may be necessary to represent the carriage return in this manner, since the computer does not recognize the depression of the carriage return key as part of an input string. This function enables you to locate a text string which begins on one line and continues on the next line while operating in the program mode (refer to EDIT $PROGRAM).

In the command format string S1 contains the substitute character. To return the substitute character to its original value, issue only the command words EDIT $SUBSTITUTE without the parameter S1 or exit string mode.

Example: Use of $SUBSTITUTE

```
10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE
70 REM SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X = 10 THEN 120
110 GØ TØ 40
120 END

EDIT $SUBSTITUTE /:/ $FIND /ØNE:70 REM/ $REPLACE /ØNE AND/ $END

TIME:  0:01

READY.

10 LET X = 0
20 LET Y = 0
30 REM INITIALIZE X AND Y
40 LET X = X+1
50 LET Y = Y+X
60 REM INCREMENT X BY ØNE AND SUM CØNSECUTIVE INTEGERS
80 PRINT "SUM ØF FIRST "; X; "INTEGERS IS "; Y
90 REM SUM FIRST TEN INTEGERS ØNLY
100 IF X - 10 THEN 120
110 GØ TØ 40
120 END
```

Observe that $SUBSTITUTE allows the combining of two lines of text by translating the carriage return into a character which will be recognized by the scan.

$TEXT

EDIT $TEXT

$TEXT, always expressed without parameters, establishes one of two scan-control modes, the text mode, which allows you to search through a file ignoring all carriage returns and line numbers.

Often there are instances when you want to locate a string extending from one line to another. By entering text mode, such strings are quickly located.

To leave the text mode and enter the program mode, type $PROGRAM (refer to EDIT $PROGRAM). If you have completed your work with string functions, type $END.

$TIME

EDIT $TIME

$TIME, expressed without parameters, will give you the current run time since initial entrance into the string mode of operation. This function may precede or follow any other string function and it does not affect the location of string pointers (if any) or the translation table.

This command gives you the central processor time only (not the terminal time) and is expressed in minutes and seconds. The central processor time is automatically given after $END.

$TRANSLATION

EDIT $TRANSLATION

$TRANSLATION is a no-parameter command which allows you to list the translation table. This table contains the characters whose initial values have been modified by the character definition functions ($MULTIPLE or $SUBSTITUTE). In addition, IGNORE, BREAK, and ABORT characters (if any) are listed.

All characters in the translation table are listed according to type. For all types, the input character is listed to the left of the equal sign and its equivalent values are on the right.

The $TRANSLATION command may precede or follow any other string function. It does not affect either the string pointers or the translation table.

Example:

```
100 A B C D E F A B C D E F
110 G,H,I,J,K,L,G,H,I,J,K,L
120 M N N N N N N N N N N N


EDIT $SUB /:/ $IGNØRE /D/ $BREAK /B C/ $MULTIPLE /AK,/ $TRANSLATIØN

SUBSTITUTE:

:=CR


IGNØRE:

D


BREAK:

INPUT B
B
SPACE
C


MULTIPLE:

A=,K


ABØRT:
```

3. Abbreviations

Both the command and control words can be abbreviated. During the preparation of a program, these abbreviations can save you time since it is not necessary to type the entire command.

Command	Abbreviation
DELETE	DEL
DUPLICATE	DUP
EXTRACT	EXT
LIST	LIS
MERGE	MER
MOVE	MOV
PAGE	PAG
RESEQUENCE	RES
RUNOFF	RUN
TEXT	TEX
WEAVE	WEA
$ABORT	$AB
$BEGIN	$BE
$BREAK	$BR
$DUPLICATE	$DU
$END	$EN
$FIND	$FI
$IGNORE	$IG
$INSERT	$IN
$LIST	$LI
$LOCATE	$LO
$MOVE	$MO
$MULTIPLE	$MU
$PROGRAM	$PR
$REPLACE	$RE
$RUNOFF	$RU
$STRING	$ST
$SUBSTITUTE	$SU
$TEXT	$TE
$TIME	$TI
$TRANSLATION	$TR
.LEFT MARGIN N	.LEF N
.RIGHT MARGIN N	.RIG N
.SPACE N	.SPA N
.BREAK	.BRE
.INDENT N	.IND N
.UNDENT N	.UND N
.CENTER N	.CEN N
.PAGE	.PAG
.SLEW N	.SLE N
.IGNORE N	.IGN N
.LINK	.LIN
.LITERAL	.LIT

4. Error Messages

This section contains a list of EDIT error messages that you might receive from the system. The descriptions list possible reasons for their occurrence.

ILLEGAL COMMAND FORMAT
USE: . . (FOLLOWED BY FUNCTION EXPLANATION)

You have not properly entered the information required for this function. You have given too few parameters (MERGE, WEAVE, DELETE, EXTRACT) or too many (RESEQUENCE, MOVE); the order is incorrect (MOVE, DELETE, EXTRACT); or you have entered alphabetic information for a function permitting only numeric input.

REISSUE COMMAND

Usually given in conjunction with another error message. This indicates that your request can probably be fulfilled if (1) the parameters are slightly modified or (2) system traffic decreases.

THESE PARAMETERS HAVE PRODUCED A 6-DIGIT LINE NUMBER

The quantity 99999 is the largest line number allowed. A resequence can produce a larger number if (1) the beginning sequence number is too large or (2) the increment is too large.

THESE PARAMETERS HAVE PRODUCED A NEGATIVE LINE NUMBER

Because of the negative increment, a line number less than zero was produced. The problem can be corrected by:

1. Choosing a larger beginning sequence number
2. Choosing a smaller block or
3. Choosing a larger increment (smaller absolute value).

INCREMENT BY ZERO

This is not an error message in the same sense as the other messages listed. If you did not intend to increment by zero, you can probably repair the damage. Do not try to add lines to your program until you have resequenced again or all lines with the same number will be replaced by only the last line with that number.

UPPER BLOCK LIMIT MUST BE EQUAL TO OR GREATER THAN LOWER LIMIT

When using MOVE or DUPLICATE you must specify a block as N1-N2, where N1 is less than or equal to N2.

N3 MUST LIE OUTSIDE THE INTERVAL (N1, N2)

When using MOVE, you must select an insert number N3 which is either less than the lower block limit N1 or greater than the upper block limit N2.

PROGRAM TOO LONG

Your program has been resequenced because you have either chosen the resequence function or your resequencing is part of the function you have chosen. In the process your program has grown to more than 6144 characters. The solution according to the function chosen is:

1. Resequence
   A. Select a smaller starting value
   B. Select a smaller increment
   C. Resequence a smaller block
2. Merge
   A. Resequence the programs in such a way that WEAVE may be used and achieve the same result.
   B. Delete one or more statements in one or more of the programs.
3. Move
   A. Delete one or more lines.
   B. Resequence only the blocks being moved.
4. Duplicate
   A. Duplicate at fewer points.
   B. Duplicate a smaller block.
   C. Remove a line in the program before duplicating.

PROGRAM TOO LONG AT XXX

During a duplicate, the size of your program increased to more than 6144 characters. The line number given is the last line at which an insertion was performed.

MODIFIED PROGRAM HAS TOO MANY LINES

A maximum of 255 lines are permitted. The MERGE, WEAVE, or DUPLICATE will not be performed unless the resulting program is within this limit.

MERGED PROGRAM TOO LONG WITH ...

The merged program must not be more than 6144 characters. If there are more than two programs in the merge list, it may be possible to merge them all by merging two at a time.

Computer Centers and offices of the Information Service Department are located in principal cities throughout the United States.

Check your local telephone directory for the address and telephone number of the office nearest you. Or write . . .

General Electric Company
Information Service Department
7735 Old Georgetown Road
Bethesda, Maryland 20014

GENERAL ELECTRIC

INFORMATION SERVICE DEPARTMENT

708209B (20M) 7-68