Following the listing of the program or routine is a list of
statement numbers and of the sequence numbers of the statements
in which these statement numbers are referenced. If renumbering
is not being performed, this list will also include as negative
numbers the sequence numbers of the numbered statements
themselves. These negative numbers are not included in the list
of statement number references if renumbering is being performed,
since then the resulting statement numbers will be in order so
that it is assumed that further assistance is not needed to
locate them.
#
After the list of statement number references is a list of all
key words, names and constants used in the program or routine and
of the sequence numbers of the statements in which these are
used. A sequence number is given as negative if the referenced
variable or array is defined in the statement by the equals sign
operator. Key words such as GO TO are considered to be single
words in the index, but blanks and tabs are otherwise used as
word separators. Except for the initial key words, items
appearing either in a DATA statement or in a FORMAT statement are
not included in the list.
#
Changing Length of Uninterrupted Listing
#
The tables of statement number references and indexes are printed
at the end of each program or routine, or when the necessary
storage fills. As supplied, RENBR will produce uninterrupted
listings of FORTRAN routines of approximately 800 non-comment
lines (dependent on programmer style). This requires 2 arrays of
1000 locations each to store the statement references, and 1
array of 7000 locations to store the symbol dictionary. If the
storage necessary to run RENBR must be reduced, it is suggested
that all appearances of the number 1000 be changed to 500 and
that all appearances of 7000 be changed to 3000. This will cause
the tables and indexes to be dumped after processing of about 250
non-comment lines, but the listings will still be correct.
#
Maintaining Logical Blocks of Statement Numbers
#
Some programmers select statement numbers used within a logical
division of a program from a different range than those used
elsewhere within the same program. RENBR can maintain these
regions when renumbering. However, since a single normal
renumbering would destroy such regions, a command card within the
program is used to specify the step size rather than querying the
user for this information. This command card is a comment card
with C in column 1 followed by the word RENBR and those switches
for which values are being specified. A typical command card
would be
#
CRENBR I-10 B 10 O 200 N 400

The switches are
#
B = the base number. The number following this switch will be the smallest generated statement number which will appear within the renumbered program or routine. Normally this would be the first statement number in the renumbered program, but if the increment between generated statement numbers is specified to be negative, then the base number will be the number of the final numbered statement within the program. If the B switch does not appear on the CRENBR card, then the value used as the base number will be that specified by the user when RENBR was started, or will be the absolute value of the statement number increment if the user did not specify a base number when RENBR was started.
#
I = the statement number increment. The number following this switch will be the increment between generated statement numbers. If the I switch does not appear on the CRENBR card, then the value used as the increment will be that specified by the user when RENBR was started, or will be 1 if the user did not specify a statement number increment when RENBR was started.
#
N = the new region size. The number following this switch specifies the jump in generated statement numbers from the start of one region to the next in the renumbered program. If the N switch does not appear on the CRENBR card, then the old region size as specified by the O switch is also used as the new region size.
#
O = the old region size. The number following this switch specifies the jump in the original statement numbers from the start of one region to the next in the program which is being renumbered. If the values specified by the N and O switches differ, then the value of the O switch should probably be converted after renumbering to that which the N switch had before renumBering. If the O switch does not appear on the CRENBR card, then no regions will be preserved.
#
The switches can appear in any order on the CRENBR card. Only the I switch will accept a negative value. Blanks or tabs can appear anywhere after the initial C of the CRENBR card, but are not necessary.
#
The values specified by the CRENBR card apply to all the statement numbers within a single program, but the CRENBR card can appear anywhere in the program before the terminal END statement. If multiple CRENBR cards appear within a single program or routine, the values used are the final values specified for each switch. The CRENBR card applies only to the program or routine in which it appears. The default values of

the switches (set by the user when RENBR was started) are
restored for the next program or routine read. In the case of
the N and O switches, these defaults are to not preserve regions.
#
The sample CRENBR card show above would specify that original
regions 1-199, 200-399, 400-599 etc. would be translated to
regions such as 1-399, 400-799 and 800-1199. The actual
translation would depend on the relative placement of the
original regions. For example, using the above sample CRENBR
specification, the statement number sequence
#
270 350 260 351 15 2 6 150 99 1600 1622
#
would be translated to the following sequence
#
830 820 810 800 440 430 420 410 400 20 10
#
To allow addition of new statements to a region, statement
numbers which would normally be outside the region are taken as
part of the surrounding region if statement numbers both before
and after the addition are within the region. Therefore, the
statement number sequence
#
270 8350 4260 351 15 2 6 150 99 1600 1622
#
would be renumbered to the same sequence as given before.
#
Statement Types Recognized by RENBR
#
RENBR will renumber statement numbers contained in the following
types of FORTRAN-II and FORTRAN-IV statements. These key words
can also be used for variable names or for array names without
causing any difficulties. The ability to handle statement
numbers in other types of statements can be easily added.
#
ACCEPT — ACCEPT TAPE
ASSIGN
CALL
DECODE
DO
ENCODE
FREQUENCY
GO TO
IF — IF ACCUMULATOR OVERFLOW
IF DIVIDE CHECK — IF QUOTIENT OVERFLOW
PRINT
PUNCH — PUNCH TAPE
READ — READ INPUT TAPE
REREAD
TYPE
WRITE — WRITE OUTPUT TAPE

In addition to the above, the FIND statement is recognized so that the unit and record numbers can be separated before being entered into the index. To prevent filling and index with often useless information, the DATA and FORMAT statements are also recognized so that items following these keywords are not indexed.
#
RENBR handles as single units alphameric strings which are designated either as a decimal number of characters followed by the letter H and the characters of the string, or else are preceded and followed by apostrophes. It is likely that no harm will result if programs containing other alphameric string designations are renumbered. If some character other than apostrophe is used as the delimiting character in programs to be renumbered, the only situation likely to give trouble will be when IF statements are used to test character data against alphameric strings containing either left or right parentheses. RENBR was used for several years before the ability to handle alphameric strings as single units was even added to it.
#
Statement numbers appearing in CALL statements can be preceded either by an asterisk, by a dollar sign or by an ampersand. The RENBR program must be changed if some other character is used to indicate such statement numbers in CALL statements.
#
Format of Statements in the Output File
#
The text of statements in the renumbered output file begins in column 7 if card format has been selected, or else following the tab which separates the statement number field from the statement text field. Original indentation in the form of additional blanks or tabs is not preserved. Where possible, in a continuation line the original number of blanks separating the statement continuation character from the text of the continued statement is maintained. All other spacings are left unchanged.
#
Files Provided
#
RENBR.F4 Source of the FORTRAN renumbering program.
#
RENBR.DAT Test data for RENBR. Directions for use are given as comments at start of file.
#
RENBR.LST This file.
#
RENBR.RNO File which generates this file when processed by the PDP-10 text justifier program RUNOFF.
#
REFMT.F4 FORTRAN program used with RENBR BLOCK DATA routine and with the DATAST package to reorder the RENBR syntax recognition table. Directions for use are

```
#                   given as comments at start of file.
#
# DATAST.F4         FORTRAN subroutine package which generates as
#                   output the FORTRAN DIMENSION, EQUIVALENCE and DATA
#                   statements which represent an input single
#                   precision singly subscripted integer array,
#                   calling sequence is described in subroutine DATA.
```

REPORT

The REPORT subsystem is used to interrogate a data base which contains a running commentary on the status of the BBN-TENEX system. The data base includes system Interruption Reports, System Restart Reports and Scheduled Downtime Reports. The program is self-explanatory. It asks the following questions:

ENTER DATE FROM WHICH TO LIST STATUS:

(The default reply is carriage return which implies earliest date of entry in data base; otherwise, enter date in form MM/DD/YY, e.g. Ø3/12/74)

OUTPUT TO:

(The default reply is carriage return which implies TTY:; otherwise, enter a file specification or LPT:)

REVERSE OR FORWARD ORDER (R OR F):

(the default reply is carriage return which implies R, i.e., reverse chronological order for output; otherwise, enter R or F)

RUNFIL

Provides a means for using a file instead of a terminal to supply an input command stream. A file is prepared containing the character stream which would be typed on the terminal to perform some desired set of operations. The stream is initially received by the EXEC but may start any subsystem, and primary input will continue to come from the file. All output is directed to the controlling terminal in the usual manner, and the resultant typescript is indistinguishable from that produced when input is from the keyboard. Optionally, output may be directed to a file other than the terminal. However, such a file may contain "errors" in echoing, that is, the source file "echoes" present in the output file may be out of phase with the output.

@RUNFIL

COMMANDS FROM FILE: name

The user supplies the name of the file from which commands are to be taken. If the name is confirmed with carriage return, output will be directed to the terminal and processing will begin. If the name is confirmed by comma, the program will ask:

OUTPUT TO FILE: name

and the name of the file to receive primary output should be entered.

The program begins by starting an EXEC in an inferior fork, with primary input and output set as determined by the dialogue above. The command file must contain EXEC commands to start any desired subsystem. The last command should leave control at the command level of the EXEC. When the end of file is reached in the command file and any processing is completed, the inferior fork(s) will be killed and control returned to the top-level EXEC.

Format of Command File

As stated, the command file contains exactly what would be typed on a controlling terminal to perform some desired set of operations. To facilitate certain operations, however, the following actions cause special interpretations by RUNFIL.

Control-^ acts as a warning character and interprets the immediately following character(s) as follows:

1) A-Z, [,\,],^,_,@

Converts the single character in this group to its control equivalent. Eg. ^C (i.e. control -^ followed by C) becomes ^C, etc.

Note: Control-C in an input stream causes an immediate termination of processing as usual. However, for some subsystems, ^C is the only way to return to the EXEC, and the user would normally not type ^C for this purpose until processing had been completed. In a command file, ^^B serves this function, i.e. RUNFIL waits until the inferior program is ready for more input (an indication that processing is completed), and then sends a ^C.

2) Decimal number

The number will be taken as a length of time (in milliseconds) to wait before taking further input from the command file.

The number must be terminated by a non-numeric character
(e.g., space), which will have no other effect in the input
stream. This "pause" facility is furnished to allow orderly
input from the terminal at desired points in the file
stream.
#
Note that once processing has begun, any characters typed on the
controlling terminal will be intermixed with the file stream,
usually producing undesired results. In particular, a ^C typed
on the controlling terminal will be handled by the lower-level
EXEC, and so is not a way to abort command file processing. The
character ^B is enabled in RUNFIL for this purpose. (^B is the
only character which has a different effect when input from the
terminal.)
#
In most cases ^B from the terminal will cause an immediate
termination of command-file processing and a return to the
top-level EXEC. However, ^B typed during a requested "pause" in
command-file input (^^<decimal number> above) will break the
pause, i.e., RUNFIL will "wake up" and immediately resume command
file input. Thus processing can be caused to continue after
manual terminal input without waiting out the remaining specified
delay. An actual ^B (as opposed to ^^ followed by B in the file
stream is treated as ordinary input and is passed to the
subsystem running under RUNFIL.

RUNOFF

RUNOFF is a TENEX program to facilitate the preparation of typed or printed manuscripts, such as memos, manuals, etc.

The user prepares his material on any regular TENEX terminal, and writes it onto a file using TECO. The user includes not only textual material, but also case and formatting information. RUNOFF then takes the file and reproduces it onto the line printer, teletype, any terminal or other file to produce a final copy or final file image. It performs the formatting and case shifting as directed, and will also perform line justification, page numbering and titling, etc., as desired.

The principal benefit of such a program is that files prepared for use with it may be edited and corrected easily. Small or large amounts of material may be added or deleted, and unchanged material need not be retyped. After a set of changes, the program may be operated to produce a new copy which is properly paged and formatted. Documentation may thus be updated as necessary without requiring extensive retyping.

TENEX Operating Procedure

RUNOFF is a regular TENEX subsystem, and is started by the EXEC command

@RUNOFF_
(In this document, _ will be used to mean carriage return.)

The program will print its name and version and ask for the input file name:

(1) INPUT FILE:

The user should type the name of the source file (using alt-mode for name recognition) and confirm the name with a carriage return or comma. If a carriage return is typed, RUNOFF will assume standard settings for all modes and immediately begin to produce output to the line printer. Typing a comma allows the user to specify various additional options as follows.

(1) INPUT FILE: name,
(Comma to ask for options)

(2) ASCII, DURA OR FLEXO?
(Type A, D, or F. If D or F typed, output will be to paper tape punch with no further questions asked.)

(3) OUTPUT FILE:
(Confirm (c) with carriage return for no more questions, comma to specify following options.)

(4) UNDERSCORE (L, C, B OR N)
(Underscore by line, character, backspacing or none. Use L (normal) for line printer, C for flexo or dura, N for anything which does not have underscoring capability.)

(5) SIMULATE FORM FEED (Y OR N)?
(Yes means use repeated line feeds to eject to top of each new page. No means use real form feed character. Use N (normal) for line printer, flexo and dura.)

(6) PAUSE?

(Answer Y or N. Yes means stop program at completion of each page so that paper can be adjusted in on-line output device. Used only for direct output to TTY. Use N (normal) for line printer, flexo or dura.)

When finished with the current input file, RUNOFF will type DONE and await specification of a new input file.

Summary of Operations

1. For output to line printer:

   Type carriage return after input file name.

2. For Dura typewriter paper tape:

   Type comma after input file name,

   Type D to question 2.

3. For Flexowriter paper tape:

   Type comma after input file name,

   Type F to question 2.

SOURCE FILE FORMAT

As stated above, the source file contains the textual material which will appear on the final copy, plus information to specify formatting. Most importantly, upper and lower case information may also be supplied so that copy can be prepared on the teletype or other such device which can normally input only upper case letters. All command information consists of regular ASCII printing characters so that a listing of the source file may be examined if the final copy is not exactly as desired.

All material in the source file is taken to be source text except those lines beginning with a period. A line beginning with a period is assumed to be a command, and must match one of those listed below. The commands provide the formatting information, and control various optional modes of operation.

Usually the text is filled and justified as it is processed. That is, the program FILLS a line by adding successive words from the source text until one more word would cause the right margin to be exceeded. The line is then JUSTIFIED by making the word spacings larger until the last word in the line exactly meets the right margin.

The user may occasionally wish to reproduce the source text exactly which is done by disabling filling and justification. The program may be set to fill but not justify, in which case the output will be normal except that lines will not be justified to the right margin. The program may also be set to justify but not fill, although this would probably produce peculiar results and is not recommended.

When the fill mode is on, spaces and carriage returns occurring in the source text are treated only as word separators. Multiple separators are ignored.

Some of the commands cause a BREAK in the output. A break means that the current line is output without justification, and the next word goes at the beginning of the next line. This occurs at the end of paragraphs.

The program will advance to new pages as necessary, placing the title (if given) and the page number at the top of each page. The user may explicitly call for a page advance where desired, and may inhibit the occurrence of a page advance within specified material.

CASE INFORMATION

Specification of case for files prepared on the teletype is done with two characters, up-arrow (↑, shift-N), and back-slash (\, shift-L). The appearance of an up-arrow causes the letter immediately following to be transmitted in upper case. The appearance of a back-slash causes the letter immediately following to be converted to lower case. Any letter not preceded by one of these characters is transmitted in the current mode. The mode is initially lower case, and is changed by the occurrence of two successive case control characters. Two up-arrows (↑↑) cause the mode to be set to upper case, and two back-slashes (\\) cause the mode to be set to lower case.

The use of the above corresponds to the use of the shift and shift-lock keys on a typewriter. Usually, typing appears in lower case. To type one letter in upper case, the shift key is used. The shift-lock is set to type a series of upper case letters, after which it is released.

The following shows the uses of the case control characters:

↑HERE IS A ↑SAMPLE ↑SENTENCE IN ↑↑UPPER CASE\\ AND LOWER CASE.

becomes:

Here is a Sample Sentence in UPPER CASE and lower case.

Note: Case conversion takes place only on ASCII codes 101 to 132 octal, that is, the upper case letters. Any actual lower case letters (codes 141 to 172 octal) appearing in the source will be transmitted unchanged. If the source is prepared on a device such as a flexowriter or model 37 teletype or any terminal which produces letters of the proper case, the mode should be set to upper case at the beginning of the file and left unchanged for the remainder.

Special Characters

The character ampersand (&, shift-6) is used to specify underscoring. The ampersand will cause the character following it to be underscored, e.g. &f&o&o becomes foo.

Underlining of a string of characters can also be specified, similarly to the use of the shift lock operations described above. An appearance of ampersand preceded by up-arrow (^&) will cause underlining of all following characters except space. An appearance of ampersand preceded by backslash (\&) will disable this mode.

It is occasionally necessary to include spaces in the text which should not be treated as word separators. For this purpose, RUNOFF treats numbersign (#) as a quoted space; i.e. it will print as exactly one space in the output, will never be expanded nor changed to a carriage return.

To allow the appearance of the special characters (ampersand, number-sign, up-arrow, or back-slash) in the output, the character left-arrow (_, shift-0) is used as a quote character. The character immediately following a left-arrow will be transmitted to the output with no formatting effect. The left-arrow itself is just another case requiring the usual quote characteristic. The following five cases occur: _&, _^, _\, __, and _#.

COMMANDS

The following commands will be recognized if they are at the beginning of a line started with a period. Any line in the source file beginning with a period is assumed to be one of these commands. If it is not, an error diagnostic will be typed and the line will be ignored. Some commands take one or more decimal numbers following. These are separated from the command by a space.

Formatting

.BREAK

Causes a break, i.e. the current line will be output with no justification, and the next word of the source text will be placed at the beginning of the next line.

.SKIP n

Causes a break after which n*(line spacing) lines are left blank. If the skip would leave room for less than two printed lines on the page (i.e. if there are less than n+2*(line spacing) lines left), the output is advanced to the top of the next page.

.BLANK n

Exactly like SKIP, except that n (rather than n*(line spacing)) lines are specified. BLANK is used where space is to be left independent of the line spacing; SKIP, where the space should be relative to the size of line space.

.FIGURE n

Like BLANK except that if less than n lines remain on the current page, the page will be advanced, and n blank lines will be left at the top of the new page. Principally used where it is desired to leave room for a figure to be drawn in manually.

.INDENT n

Causes a break and sets the next line to begin n spaces to the right of the left margin. n may be negative to cause the line to begin to the left of the left margin (useful for numbered paragraphs).

.PARAGRAPH n

The number is optional and, if present, sets the number of spaces which paragraphs are to be indented. The initial setting is 5. The command causes a break and leaves (m+1)/2 blank lines, where m is the regular line spacing. The next line will be indented as indicated above.

.PAGE

Causes a break and an advance to a new page. Does nothing if the current page is empty. Titling and numbering as for automatic page advance.

.TEST PAGE n

Causes a break followed by a conditional page advance. If there are n or more lines remaining on the current page, no advance is made and no lines are skipped. Otherwise, the page is advanced as for PAGE. This command should be used to ensure that the following N lines are all output on the same page.

.NUMBER n

Turns on page numbering (normal) and, if n is supplied, sets the current page number to n.

.NONUMBER

Turns off page numbering. Pages will continue to be counted, so the normal page number will appear if numbering is re-enabled.

Mode Setting

.JUSTIFY

Causes a break and sets subsequent output lines to be justified. (Initial setting)

.NOJUSTIFY

Causes a break and prevents justification of subsequent output lines.

.FILL

Causes a break and specifies that subsequent output lines be filled. Sets the justification mode to be that specified by the last appearance of JUSTIFY or NOJUSTIFY. (Initial setting)

.NOFILL

Causes a break and prevents filling of subsequent output lines. Also turns off justification.

Note:

1. The nofill-nojustify mode need be used only where there are several lines of material to be copied exactly. A single line example will not require using these commands if there are breaks before and after.

2. Normally FILL and NOFILL are used to turn both filling and justification on and off. It is usually desirable to do both. A subsequent appearance of a justification command will override the fill command however.

3. Because of the action of FILL, a single occurrence of NOJUSTIFY will cause the remainder of the file to be unjustified, with filling as specified. In order to justify but not fill (not recommended), a JUSTIFY command must follow every NOFILL command.

Parameter Settings

.LEFT MARGIN n

Causes a break after which the left margin is set to n. n must be less than the right margin, but not less than Ø. The initial setting is Ø. The amount of any indent plus the left margin must not be less than Ø.

.RIGHT MARGIN n

Causes a break after which the right margin is set to n. n must be greater than the left margin. The initial setting is 6Ø.

The number of characters on a line will be equal to or less than the right margin minus the left margin minus any indenting which may be specified. Even if filling has been disabled, lines will not be extended past the right margin.

.SPACING n

Causes a break after which the line spacing will be set to n. n must be within the range 1 to 5. Single spacing is 1, double spacing is 2, etc. (Initial setting is .SPACING 2)

.PAGE SIZE n

Sets the number of lines per page to n. n must be greater than 1Ø. The initial setting is 58. n includes the top margin of 5 lines. The page number and title appear on the third line.

.TAB STOPS n ... n

Clears all previous tab stops and sets new tab stops as specified. The several n (max 32) must be greater than zero and in increasing order. They are the positions of tab stops independent of the setting of the left margin, although any which are less than the left margin will not be seen. There are no tab stops initially.

Tabs should be used only in lines which will be unjustified and unfilled, i.e. where filling is disabled or a break immediately follows. Clearly, the spaces specified by a tab character should not be expanded to justify the line--this would defeat the effect of tab formatting. The appearance of a tab in the source text will be translated to one or more spaces, the amount necessary to advance to the next tab stop. If a tab appears at a point where no further tab stops have been set on a line, the tab will be treated as

though it had been a space.

Miscellaneous

.TITLE tttt ... tttt

This command takes the remaining text on the line as the title. This text will appear at the top of all subsequent pages, at position Ø, on the third line with the page number. The title is initially blank.

.SUBTITLE tttt ... tttt

This command takes the remaining text on the line as the subtitle. This text will appear on the line immediately following the title and page number. The subtitle is initially blank. The subtitle is not indented, but may contain leading spaces to achieve the same effect, if desired.

.CENTER

This command causes a break after which it centers the next line following in the source file. The centering is over position 3Ø, independent of the setting of the left and right margins.

.FOOTNOTE n

Allocates n*(line spacing) lines at the bottom of the current page for a footnote(1). If insufficient room remains on the current page, space will be allocated at the bottom of the following page. The text for the footnote should begin on the line following the command, and it may contain any appropriate commands (e.g. CENTER, SKIP) necessary to format the footnote. The footnote is terminated by a line beginning with an exclamation point (the remainder of which is ignored). The lines delimited by this line and the FOOTNOTE command are put into a buffer to be processed when the output moves to within the stated distance of the bottom of the page. If a page has multiple footnotes, the allocated space is the sum of the allocations for all footnotes assigned to the page. The user must

- - - - - - - - - - -

(1) This is a footnote. This text and the dividing line above were specified by text and commands following a FOOTNOTE 5 command.

include his choice of footnote-designating symbols within the text.

The current values of left and right margin and line spacing are saved and restored after processing of footnotes. Therefore, a footnote may contain commands which change these parameters, and the effect will be limited to the footnote text.

The actual space taken by the footnote may be more or less than that specified by n. The n merely allocates space and should be the user's best guess. If it is considerably off, the footnote lines may overflow the page, or extra space may be left at the bottom. The user may wish to adjust n after examining a first draft printout.

.INDEX tttt ... tttt

This command takes the remaining text on the line as a key word or words and adds it, along with the current page number, to the internal index buffer. The command does not cause a break. It should appear immediately before the item to be indexed. A key word or words may be indexed more than once.

.PRINT INDEX

Causes a break after which it prints the entire contents of the index buffer. Entries are printed in alphabetical order, and are set against the left margin. Regular line spacing is used, except that a blank line is left between entries of different first letters. The number of the first page on which each entry appeared is put on the same line as the entry, beginning at the middle of the line (midway between the left and right margins). Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

INDEX
(Entries entirely in upper case are command names.)

SNDMSG

SNDMSG is used to send messages to users throughout the ARPANET. It prompts for the addresses to which to send the message, then for the subject and text of the message. It automatically fills in the date and who the message is from. Finally it sends the message. If the message can't be delivered immediately, it is queued and automatically sent later by the background MAILER process.

The following sections describe the various stages of SNDMSG (telling how to enter each kind of information) and the form of the message that is actually sent.

TO SUPPRESS TYPEOUT FROM SNDMSG, TYPE ^O (CONTROL-O) WHILE IT IS TYPING.

A. The Addresses

Addresses are divided into "To" and "cc". SNDMSG prompts separately for each. When SNDMSG prompts for addresses, enter them separated by commas. Carriage return terminates the list and goes on to the next section unless it is preceded by a comma, in which case address input continues on the next line. There must be at least one "to" address, but there needn't be any "cc". If you don't enter any "To" addresses, SNDMSG will skip to the subject and message, then will come back to the addresses.

The following kinds of items may appear in the address list separated by commas. (angle brackets are not typed - they are just used in this description to enclose a type of item)

1) <user name> - The mail is addressed to that user at the default host.
   Normally the default host is the local host (i.e. the one on which you are running SNDMSG) but you can change it as desired (see below).

2) @<Host name or octal host number>
   Changes the default host to be the given host. The default remains in effect for the rest of the address list until explicitly changed. To set the default back to local, type just "@" (i.e. omit the host name)

3) <user name>@<host name or octal host number>
   The mail is addressed to the given user at the given host. Putting the host name together with the user name makes it apply only to that user. That is, it does not change the default host, but overrides it just for the user. A blank host name means the local host.

4) <Group name>: - All subsequent addresses will be considered part of the specified group (which can be any name you care to define). This group remains in effect until explicitly changed by another "<group name>:" item. Typing just ":" with no name in front sets the group to none - i.e. subsequent addresses are ungrouped. Group names do not affect the sending of the message (they are not addresses), just what appears in the heading of the transmitted message (see section E).

(Note: It is not necessary to separate the group name from the next address by a comma - the colon implies a comma.)

5) *<File name> - The message will be sent to the given file. Files are always local and must already exist. Defaults for fields you omit are:

directory: logged in directory (not connected)
name: SAVED
extension: MESSAGES
version: highest existing

File name addresses are omitted from the message heading (see section E).

Special characters for editing, etc. are as follows:

1) recognition

altmode (ESC) - If typed part way through a user or host name, causes recognition of the name if possible. The remainder of the name is typed out. If the name typed so far is ambiguous, a bell is rung.

2) correcting mistakes

There is no way to edit anything but the current item being typed (i.e. editing won't go past the preceding comma). The following control characters edit the current item or kill the whole address list.

^A - deletes preceding character - will only go back as far as preceding "@", ",", ":", or "*".

^H - same as ^A

^Q - deletes current word

^W - same as ^Q

^X - deletes whole address list, starts over

```
#       rubout - deletes current item (back to "," or ":")
#
#    3) Viewing addresses
#
#       ^R - retypes current word
#
#       ^S - retypes whole address list
#
#    4) Inserting a file
#
#       A file containing addresses may be prepared ahead of time
#       and inserted into the SNDMSG address list. Addresses are
#       entered on the file exactly as they would be typed to SNDMSG
#       except that carriage returns from a file will not terminate
#       the address list - thus addresses on a file may appear on
#       separate lines without commas if desired.
#
#       To insert the file, type ^B (control-B) during SNDMSG
#       address input. You will be asked for a file name (which you
#       type in the usual TENEX way) and for confirmation (type
#       space or carriage return to confirm). The file is read into
#       the address list exactly as if you had typed its contents.
#       When SNDMSG types EOF (End-of-file) you continue entering
#       addresses as usual.
#
#       Note: Although files are often used to hold definitions of
#       groups (i.e. they often start with a "<group name>:"),
#       files and groups are two distinct features. The purpose of
#       a file is to permanently hold some list of addresses. The
#       purpose of a group name is to prevent the component
#       addresses from cluttering up the message heading. Although
#       groups may be defined in files, they may also be typed
#       directly into SNDMSG, and files inserted into SNDMSG need
#       not define groups.
#
# B.  The Subject
#
# When SNDMSG prompts for "Subject", type in up to one line,
# terminated by carriage return. You may type just carriage return
# to omit having a subject. If a subject is entered, it appears in
# the heading of the message. The same control characters
# described below under "Message" (section C) apply to the subject
# as well, with the following additions and exceptions.
#
# ^Z has no special meaning
# ^B automatically means insert file, never invoke TECO
# ^X deletes the whole subject and starts over
# rubout is same as ^X but echoes differently.
```

C. The message
#
When SNDMSG prompts for "Message" type in the text of the message. When you are done composing the message, type ^Z (control-Z) to finalize it and to go on to the next stage.
#
Various control characters invoke features that aid in composing the message. They are:
#
1) Correcting typing mistakes
#
^A - deletes previous character
^H - same as ^A
^Q - deletes current line (if typed at beginning of line, deletes previous line)
^W - deletes previous word
#
2) Viewing the message
#
^R - retypes current line
^S - retypes whole message
#
3) Inserting pre-composed text and editing
#
Especially when dealing with large messages, you may wish to prepare your message ahead of time on a file or you may find the local editing characters above insufficient. If you type ^B (control-B) SNDMSG asks whether you want to insert a file or invoke TECO (a general text editor).
#
If you answer "T" TECO is started up with your message in its buffer. When you exit TECO (with ";H") the edited message is passed back to SNDMSG and you continue composing the message as usual (you are positioned at the end of the message, as if you had just typed it into SNDMSG).
#
If you answer "F" SNDMSG asks for the name of the file to insert. Type the file name in the usual TENEX way (the usual editing characters and recognition apply). You will then be asked to confirm it. If you do (for example, you type space or carriage-return) the file is read in and appended to any text you have already entered, and "EOF" is typed. You then continue message composition as usual. NB: Characters read from the file are treated just like teletype input, so be sure the file doesn't contain any of the control characters which have special meanings in SNDMSG!
#

MESSAGE.COPY
#
When you terminate input (^Z) or invoke TECO, the message is saved on the scratch file MESSAGE.COPY in the connected directory. Thus if you mess up your text in TECO or simply decide not to send the message yet you can quit SNDMSG (^C) and run it later, inserting the copy with ^B. Note that since MESSAGE.COPY is a scratch file, it is deleted when you log out!
#
CAVEAT - The message must not contain a line consisting of the single character "." if it is to be sent over the network.
#
#
#
D. Sending the message
#
When all information has been entered (addresses, message, etc.) SNDMSG asks whether to send the mail now or queue it for later delivery (the question is: "Q,S,?,carriage-return"). Type "?" for an explanation of the commands. If you just answer with carriage return, the message is sent immediately. The queueing or sending begins after this command is given -- i.e., when you terminate it with carriage return. (Note: If you have addressed the message to a file using * in the address list, SNDMSG will try to deliver it even if queueing is in effect.) The message is queued for or delivered to each of the addresses. For local addresses (ones in which no host was specified) duplicates are eliminated.
#
SNDMSG types each address as it starts to process it, then the outcome. There are three possible outcomes:
#
1. "ok" - the message has been successfully delivered to the address.
#
2. "queued" - the message was not delivered, but has been queued for later delivery. If SNDMSG was supposed to deliver the message (because of the commands you gave) the reason it had to be queued is typed. The message is placed on the file [--UNSENT-MAIL--].address in the directory under which you are logged in. MAILER, running in the background, will deliver it a soon as possible (see writeup of MAILER). In some cases (SNDMSG will tell you if this is the case) you may have to run MAILER yourself.
#
3. "can't" - the message could not be delivered, and SNDMSG did not queue it because the reason for the failure looked permanent. The reason is typed out, and the message is placed on the file /UNDELIVERABLE-MAIL/.address in the directory under which you are logged in. Thus, for example, if the problem was that you misspelled the person's name,

```
#       you can correct it and requeue the message without having to
#       compose  it all over again.  (See the writeup of MAILSTAT in
#       this manual.)
#
# For an explanation of what happens to queued mail see the writeup
# of MAILER.
#
# For information on checking the status of  queued  mail  see  the
# writeup of MAILSTAT.
#
# E.  What the message looks like
#
# The message has the following form:
#
# Date: <date and time the message was composed>
# From: <name under which you were logged in when you sent it>
# Subject: <whatever you typed - omitted if you didn't give a subject>
# To: <all the addresses you send the message "To" except for file
# (as opposed to user) addresses - addresses entered as part of a
# group are replaced by the group name>
# cc: <same as "To" but for the "cc" addresses - omitted if none>
#
# <the message>
```

SNOBOL

The TENEX SNOBOL is identical to the one distributed by DEC with the following exception:

> Vertical Tab and Form Feed characters in a user's data will be passed to his program like any other character, instead of faking an end-of-line.

SNOBOL users may obtain the following excellent references:

1) Wade, L.P., "SNOBOL4 Version 3.4", Digital Equipment Corporation, DECUS no. 10-104, Nov. 24, 1970.

2) Griswold, Poage, Polonsky, "The SNOBOL4 Programming Language", Prentice - Hall, 1968.

SORT

SORT is a column-oriented text file sorter. It is described in Appendix E of the DEC COBOL manual, and also under the COBOL section of this User Guide.

Example:

To sort a file called SORT.IN which contains the sorting information in columns 5 through 10 of each line, and where each line is no more than 72 characters, type the following:

```
@SORT
*SORT.OUT/A/R72_SORT.IN/K5.6
SORTED 423 RECORDS
comment %
```

SON OF STOPGAP

N O T E T O U S E R S

This manual has been updated to reflect changes made at Utah which include Tenex implementation, removal of bugs, and new commands. It has been updated through program revision V1.2 which was the SOS version in existence at Utah from October 1972 up to October 1974. Since then, SOS has been using a Help-command and a helpfile called SOS.SUPPLEMENT to document revisions and enhancements. The reader of this file SOS.MANUAL should be aware therefore that certain aspects of SOS have changed or been extended, and that the SUPPLEMENT or, equivalently, the Help command is the up-to-date reference guide... but only a guide or quick summary.

In particular, for new users, the FILE= syntax herein has been changed to ask for just an input file or a <cr> deferral. All other changes are essentially "upwardly compatible" with this manual, but the user should browse through the SUPPLEMENT (via the Help command) to check recent revisions and new command-options.

ABSTRACT

SOS is a line-number oriented editor for text files. It features two flavors of intraline editing (for Teletypes and displays), string search and substitution, hyphenless text justification, and other glories.

CREDITS

The original STOPGAP text editor was designed and programmed by Bill Weiher. Steve Savitzky added text justification (JU, JC, JR, and JL commands) and more extensive string search features. Dan Swinehart subsequently added display line editor commands (Z and Q) and automatic file saving (SAVE and ISAVE). This manual has been rewritten by Les Earnest, and at Utah by Kevin Kay.

1. INTRODUCTION

SOS provides the ability to insert, delete, modify, and print lines of text. While most commands are line-number oriented, string search and substitution commands are available. Commands are discussed below roughly in order of increasing complexity. It is suggested that you begin by reading Sections 1 and 2, then do some text editing. Successive sections describe more elegant functions and may be consumed one at a time.

A command to the editor consists of one or two characters followed by a list of arguments. The input format is free field i.e., spaces are ignored except that they delimit numbers and identifiers. Tabs are treated as multiple spaces.

1.1 OPERATION

To start SOS, simply type SOS to the Exec's @. Alternately, there are provisions within LISP, RLISP, REDUCE, and SAIL, to link directly into the SOS editor at need.

When SOS is started from Exec-level, it will inquire 'FILE='. You may reply in any of three different ways, depending upon your immediate purpose:

1) If you want to read an existing file, then simply type in the 10X filename, followed by a return; e.g., <SOMEONE>FOO.BAH<cr> .
2) If you want to create a new file, and name it now (rather than later), then type an altmode followed by the desired name, and a return. The name will be remembered at W- or E-time.
3) If you want to create a file, but prefer to defer naming it until later (the first World or End command), then reply with just a return.

When SOS is ready to accept a command, it will type '*'. If FOO isn't found or is illegally specified, SOS will say so, then re-type 'FILE=' and expect you to give another name.

If the file name includes a directory name other than the one you are connected to, the original file will not be changed, but a new version will be created in your area. Similarly, if the file being read is on Dectape, it will not be deleted by SOS (since it can't be undeleted in the event of error).

If you wish to read a file but not modify it, you may type ',R' after the input filename, which puts you in read-only mode. In this case, any attempt to modify the file will give a non-fatal error message.

1.2 FILE HANDLING

SOS currently works by recopying the text of your file with corrections and additions onto a file whose name it invents. The name of this file is of the form EDjj.TMP, where jj is your current job number. When the edit is finished, SOS renames this temporary file to the name of the user's original file unless a new name is specified (see the E command).

If the user attempts to reference a line which occurs earlier in the file than the one last referenced, SOS may have to finish the current copy and start copying over again. This process is automatic but does take time, so there may be a delay in executing certain commands. Notice that the form of editing used means that if the user calls the system without saying either "W" or "E", his original text file will be unchanged, so any editing will have been lost.

SOS keeps a portion of the file in core during the editing process so that a certain amount of backup can be done without recopying. SOS will use all the core it is given for this purpose. However, note that running SOS in a large amount of core may result in much poorer response time.

There are certain error messages which may occur at almost any time during an edit and which cause the editor to call exit. Note that this will cause the edit to be lost, but the original file will not be harmed. These errors are as follows:

DEVICE OUTPUT ERROR Output error. No recovery will be attempted.

DEVICE INPUT ERROR Input error. No recovery will be attempted.

INTERNAL CONFUSION or ILLEGAL UUO or TENEX CONFUSION
SOS has just discovered a bug in itself.

In addition to the above, there are two messages which are primarily warning messages. The first of these is *LINE TOO LONG*. This means that some line in your text is too long (more than 147 characters).

The line will be shortened to 147 characters and printed. The second message is *OUT OF ORDER*, which indicates that some line of your input has a number lower than the line before it. The line which is out of order will be printed. In both of the above cases, the page number on which the error occurs will also be printed.

The error *ILLEGAL LINE FORMAT* may occur at any time. This probably means that you have a line containing a <return> not followed by a <line feed> or a <line feed> not preceded by a <return>. The best cure is to replace the line completely (R command).

1.3 SPECIFYING LINES AND RANGES

The text file is organized in terms of pages which are subdivided into lines. The pages are numbered sequentially starting at 1 and continuing to the end of the file. The division into pages is determined by the user rather than by the editor. Page marks (which indicate the start of a page) may be inserted and deleted by the user. Page numbers are "floating"; that is if page mark 2 were deleted, the page which was formerly numbered 3 would now be page 2.

In contrast to this each line on a given page has a line number which is "sticky". That is, a given line will retain the same number regardless of insertions or deletions. The lines are usually numbered by 100 or some other increment larger than 1 to allow room to insert new lines. Several pages may have lines of the same number.

Most SOS commands refer to either a single line or a range of successive lines. A single line is specified by giving both the line number and the page number in the form <line number>/<page number> as 100/3 for line 100 on page 3. A range is specified by giving the first and last lines of the range separated by a colon as 100/3:4702/6 for line 100 on page 3 through line 4702 on page 6. Alternatively, a range may be specified by a starting line and a number of lines. For example, 200/4!10 refers to the 10 lines beginning on line 200 of page 4. In either case, this construction is referred to as a range specifier.

Instead of a number the symbol "." may be used. "." means either the current page or the current line depending on whether it appears after or before the /. Thus 100/. means line 100 on the current