.ds Double space: begin double spacing the text. This control line is equivalent to **.ms 2.** This control line causes a break.

.dt name n
Define a table: **Name** will be defined to be a table of length **n.** The elements of **name** will be initiaized to zero (not the null string) and can be assigned values by means of a .st control line. The elements can be accessed by use of the **%name|nnn%** construct. The indexes for the table range from zero to **n-1.**

.eh # ttl
Even footer: this defines even page footer line number **#.** If the title line is omitted, the footer line with that number is cancelled.

.eh # ttl
Even header: this defines even page header line number **#.** If the title line is omitted, the header line with that number is cancelled.

.eq n Equation: the next **n** text lines are taken to be equations. If n is omitted, 1 is assumed. This ccntrol line implies **.ne %Ms%*n** so that all equations will be on the same page. Each equation shold be formatted as a title line.

.fh ttl
Footnote header: before the first footnote on a page is printed, a demarcation line is printed to separate it from the text. The format of this demarcation line is specified by the **ttl.** The default footnote header is a line of underscores from column one to the right margin.

.fi Fill: this control line enables fill mode. This is the default condition. This control line causes a break.

.fl c, n
Flag control: The character **c** is set up as the flagging character. If **n** evaluates to be negative or zero flagging will be turned off, otherwise a flag will be printed in column **n** of every output line.

.fo # ttl
Footer: even and odd footers are set at the same time. This control line is equivalent to:

```
.ef # ttl
.of # ttl
```

.fr c Footnote reset: this control line specifies footnote numbering according to the argument **c**. Permissible values of the argument are:

- t Footnote counter is reset at the top of each page. This is the default condition.
- f Footnote counter runs continuously through the text.
- u Suppress numbering the next footnote. That is, the next closing **.ft** line will not change the value of **Foot**.

If any other value for **c** is supplied, **f** is assumed.

.ft Footnote: when **.ft** is first encountered, all subsequent text until the next **.ft** line is treated as a footnote. Any further text on the **.ft** line will be ignored. If a footnote occurring near the bottom of a page will not fit on the page, as much as necessary will be continued at the bottom of the next page. If a .ft line occurs when there are fewer than two lines remaining on the current page, the last text line on the current page will be removed, the page will be terminated, and the text line will be written as the first line on the next page.

.gb xxx
Go back: the current input file is searched from the beginning until a line of the form **.la xxx** is found. Processing is continued with the line following the matching **.la** control line.

.gf xxx
Go forward: same as **.gb** except that the search begins at the current position in the input file.

.he # ttl
Header: even and odd header lines # are set at the same time. This control line is equivalent to:

```
.eh # ttl
.oh # ttl
```

.if f exp
Insert file: the indicated file is inserted into the text at the point of the **.if** control line. The default directory is the currently connected one, and the default extension is ".MRN"; it is wise to cultivate the habit of always including <directory> in the file name. The inserted file may contain both text and control lines. No break occurs. Insertions may be nested to a maximum depth of thirty. If **exp** is provided, it will be evaluated and its value and type will be assigned to the variable **Parameter**; otherwise, the value of **Parameter** remains unchanged.

.in +n **I**ndent: the left margin is indented **n** spaces by padding **n** leading spaces on each line. The right margin remains unchanged. If **n** is omitted, **Ø** is used. If **n** is preceded by a plus sign or a minus sign, the indentation is changed by **n** rather than reset. This control line causes a break.

.la xxx

Label: defines the lable **xxx** for use as the target of either **.gb** or **.gf** control lines.

.li n **L**iteral: this request causes the next **n** lines to be treated as text, even if they begin with ".". If **n** is omitted, 1 is assumed.

.ll +n **L**ine length: the line length is set to **n**. The left margin stays the same, and no break occurs. The default for **n** is 65 both initially and if **n** is omitted. If **n** is preceded by a plus sign or a minus sign, the line length is changed by **n** rather than reset.

.lm +n **L**eft **m**argin: this control line has the same effect as a **.in +n** control line.

.ma +n **M**argins: top and bottom margins are set to **n** lines. If **n** is preceded by a plus sign or a minus sign, the margin is changed by **n** rather than reset. The margin is the number of blank lines left above the first header and below the last footer. The default is four lines. This control line is equivalent to

.m1 +n
.m4 +n

.mc name

Make counter: defines **name** to be a counter variable. **Name** will be initialized to one. If **name** is reset (by a **.sr** control line), it will retain its counter property so long as it is only reset to a numeric value. When a counter variable is referenced its value is returned first, and then incremented.

.mp +n **M**ultiple pages: format the output text so that it prints on every nth page. The defaut value is 1.

.ms +n **M**ultiple space: begin spacing the text so as to leave (**n**-1) blank lines between text lines. If **n** is preceded by a plus sign or a minus sign, the acing is changed by **n** rather than reset. If **n** is not given, 1 is assumed. This control line causes a break.

.m1 +n Margin 1: the number of blank lines left above the first header is set to **n**. If **n** is preceded by a plus sign or a minus sign, the margin is changed by **n** rather than reset. The default is 4.

.m2 +n Margin 2: the number of blank lines left between the last header and the first line of text is set to **n**. If **n** is preceded by a plus sign or a minus sign, the margin is changed by **m** rather than rset. The default is 2.

.m3 +n Margin 3: the number of blank lines left between the last line of text and the 1st footer is set to **n**. If **n** is preceded by a plus sign or a minus sign, the margin is changed by **n** rather than reset. The default is 2.

.m4 +n Margin 4: the number of blank lines left beteen the first footer and the bottom of the page is set to **n**. If **n** is preceded by a plus sign or a minus sign, the margin is changed by **n** rather than reset.

.na No adjust: justification mode is disabled. This control line causes a break.

.ne n Need: a block of **n** lines is needed. If **n** or more lines remain on the current page, text continues as before; otherwise, the current page is ejected and text is continued on the next page. No break occurs unless it is necessary to advance to a new page.

.nf No fill: fill mode is disabled, so that a break is caused after every text line. Adjustment mode is ignored, and adjusting is not performed. Upon a .fi control line, adjustment will continue if it is enabled at that time.

.of # ttl
Odd footer: this defines odd page footer line number **#**. If the title line is omitted, footer line **#** is cancelled.

.oh # ttl
Odd header: this defines odd page header line number **#**. If the title line is omitted, header line **#** is cancelled.

.op Odd page: the next page is forced to be odd by adding one to the page number counter if necessary. A break occurs and the current page is ejected.

.pa +n Page: The page number counter is set to **n**. If **n** is preceded by a plus sign or a minus sign, the counter is changed by **n** rather than reset. A break occurs and the current page is ejected.

.pc n Picture with caption: if **n** lines remain on the present page, then the text and commands up to the next .pc control line are immediately processed, after which normal text processing continues without a break occurring (independent of anything that might appear within the body of the caption). Otherwise, the text and commands up to the next .pc control line are queued up and will be processed when the next page is reached.

.pi n Picture: Set aside a blank space **n** lines long. This control line is equivalent to:

```
.pc n
.sp n
.pc
```

.pl +n Page length: the page length is set to be **n** lines. The default is 66. If **n** is preced by a plus sign or a minus sign, the page length is changed by **n** rather than reset.

.rd Read: one line of input is read from the controlling terminal. This input line is then processed as if it had been encountered instead of the .rd control line. Thus, it may be either a text line or a control line. A break occurs only if the processing of the terminal line causes one.

.rm +n Right margin: this control line has the same effect as a **.ll +n** control line.

.ro Roman numerals: establishes that whenever the page counter is substituted for a **%** (not for **%Np%**) it will be done with Roman, rather than the default Arabic, numerals.

.rt Return: terminates processing characters from the current input file and continues processing from the line following the .if control line of the previous input file.

.sk n Skip: **n** page numbers are skipped before the new page by adding **n** to the current page number counter. No break in the text occurs. This control line may be used to leave out a page for a figure. If **n** is omitted, 1 is assumed.

.sp n Space: space **n** lines. If **n** is not given, 1 is assumed. If not enough lines remain on the current page, footers are printed and the page ejected, but the remaining space is not carried over to the next page. However, if MRunoff is already at the top of a new page (even though it may not have printed the headers on that page yet), the space will be left immediately after the headers are printed. This control line causes a break.

.sr name exp
Set reference: associate the value (and type) of **exp** with the variable **name**. **Name** may be either a user-defined identifier or one of the built-in symbols.

.ss Single space: begin single spacing text. This control line is equivalent to **.ms 1**. This is the default condition. This control line causes a break.

.st name n,exp
Set a table entry: Entry **n** of table **name** will be set to **exp**. Notice that the comma is required.

.tb +n,
Tab stops: Sets a tab stop at the position indicated by the value of each expression in the remainder of the control line. If an expression is preceded by a plus sign or a minus sign, the expression will specify an offset from the previous tab stop location.

.tr cd Translate: the nonblank character **c** is translated to **d** in the output. An arbitrary number of **cd** pairs can follow the initial pair on the same line without intervening spaces. An unpaired **c** character at the end of the line will cause **c** to be translaaed to a blank. (translation of a character to a blank in the output is useful for preserving the identity of a string of characters, so that the string will not be split across a line, nor have padding inserted within it.)

.ts n Test: the next input line will be ignored if the value of **n** equals zero (false). The default value is non-zero.

.ty xxx
Type: write **xxx** onto the ccntrolling terminal. Substitution of variables may occur if the first or second character of **xxx** is **%**.

.uc c Underscore character: establishes the character to be used to indicate underscoring. If **c** is omitted, underscoring is completlely disabled and no character is intercepted for this purpose. This is the default mode.

.un n **Un**dent: the next ouput line is indented **n** spaces less than the current indentation. Adjustment, if in effect, will occur only on that part of the line between the normal left indentation and the right maagin. If **n** is not specified, its value is the current indentation (i.e., the next output line will begin at the left margin). This control line causes a break.

.ur xxx

Use reference: xxx will be scanned, and any indicated variable substitutions will be performed. The line thus constructed is then processed as if it had been encountered in the original input file (e.g., it may be another control line - even possibly aaother **.ur**).

.wt **Wait:** read one line from the controlling terminal and discard it.

.* This line is treated as a comment and ignored. No break occurs.

.~ This line is treated as a comment and ignored. However, the line is copied into the .CHARS file.

Control Line Summary

The following conventions are used to specify arguments on control lines:

c	a single character
cd	a sequence of character pairs
exp	expression (either numeric or string)
#	integer constant
n	numeric expression
+n	+ or - indicates update by n, otherwise set to n
f	file name
t	title line ('part1'part2'part3')
+n,	sequence of +n expressions separated by commas
sym	variable name
str	string valued expression

An asterisk at the beginning of "Meaning" indicates that the control line causes a break.

Request	Default	Meaning
(blank line)		*Equivalent to **.sp** 1
(form feed)		*Equivalent to **.bp**
.ad		*Right justify text
.ap sym str		Append **str** to **sym.**
.ar		Arabic page numbers
.bp		*Begin New page
.br		*Break, begin new line
.cc c	%	Change special character to c
.ce n	n=1	*Center next n lines
.ch cd		Flag **c** in .CHARS files as **d**
.ds		*Equivalent to: .ms 2
.dt **sym n**		Define sym to be a table of length n.
.ef # t		Define even footer line #
.eh # t		Define even header line #
.eq n	n=1	*Next n lines are equations
.fh t	line of underscores	Format of footnote demarcation line
.fi		*Fill output lines
		.fl c,n Set to flag output with a **c** in column **n.**
.fo # t		Equivalent to: .ef # t .of # t
.fr c	t	Controls footnote numbering: "t": reset each page; "f": continuous numbering; "u": numbering suppresed for next footnote
.ft		Delimits footnotes.
.gb xxx		"go back" to label xxx
.gf xxx		"go forward" to label xxx

Control Line Summary

Request	Default	Meaning
.he # t		Equivalent to: .eh # t .oh # t
.if f exp		File **f** inserted; value of **exp** assigned to **Parameter**
.in +n	n=0	*Indent left margin n spaces
.la xxx		Define label xxx
.li n	n=1	Next n lines treated as text
.ll +n	n=65	Set line length to n
.lm +n	n=0	*Indent left margin n spaces
.ma +n	n=4	Set top and bottom margins to n
.mc sym		Define sym to be a counter.
.mp +n	n=1	n-1 blank pages between output pages
.ms +n	n=1	*Multiple space of n lines
.m1 +n	n=4	Margin above headers set to n
.m2 +n	n=2	Margin between headers and text set to n
.m3 +n	n=2	Margin between text and footers set to n
.m4 +n	n=4	Margin below footers set to n
.na		*Do not right justify
.ne n	n=1	Need n lines; begin new page if not enough remain
.nf		*Don't fill output lines
.of # t		Define odd footer line #
.oh # t		Define odd header line #
.op		*Next page number is odd
.pa +n	n=+1	*Begin page n
.pc n	n=1	If n lines remain on the page, continue processing text and ignore the next .pc control line. Otherwise, save all the text and commands up to the next .pc control line and process it at the top of the next page.
.pi n	n=1	Equivalent to: .pc n .sp n .pc
.pl +n	n=66	Page length is n
.rd		Read one line of text from the terminal and process it
.ro		Roman numeral page numbers
.rm +n	n=65	Set line length to n
.rt		"Return" from this input file
.sk nn=1		Skip n page numbers before next new page
.sp n	n=1	*Space n lines
.sr sym exp		Assign value of exp to variable named sym
.ss		*Equivalent to: .ms 1
.st sym n,exp		Set entry n of table sym to be exp.
.tb +n,		Set a tab stop at each n in the list
.tr cd		Translate c into d on output

Control Line Summary

Request	Default	Meaning
.ts n	n=1	Process next input line only if n is non-zero.
.ty xxx		Write **xxx** onto the terminal
.uc c	none	Sets underscore control char; if c is omitted, underscoring disabled.
.un n	left margin	*Indent next text line **n** spaces less
.ur xxx		Substitute values of variable in **xxx** and then process **xxx**.
.wt		Read one line of text from the terminal and discard it
.*		Comment line; ignored
.~		Comment line; ignored, but written to .CHARS file

Built-in symbols

Only those symbols marked with an asterisk before their value are settable by the user. All symbols are numeric unless they are specified to be a string or a table. Control words and control arguments which afffect the values of the variables are indicated in parentheses: (x/y) indicates that x sets the switch to true (-1), and y sets if false (Ø); (a) or (a,b,c) indicates that it is affected by a, or by a, b and c.

Symbol	Value
Ad	*Adjust (.ad/.na)
AskHyphenations	*User is to be interrogated about word hyphenations.
CASECONTROL	*ASCII code of character signalling case conversion
Ce	*Number of lines remaining to be centered (.ce)
CHANGECASE	*Overall mode is to interchange upper- and lower-case letters.
CharsTable	*Translation table for .CHARS file [table] (.ch)
Charsw	*".CHARS" file is being created (character option)
Console	Reads one line from the terminal [string]
Date	Date of this invocation of MRUNOFF; format is mm/dd/yy [string]
Eq	*Equation line counter (.eq)
EvenFlagCol	*Column in which to flag even pages (.fl)
ExtraMargin	*Indent entire text this many spaces (margin option, indent option)
Fi	*Filling (.fi/.nf)
FileName	Name of primary input file [string]
Filesw	Output is going to a file (output option)
FlagChar	*ASCII code of character to be used to flag output lines (.fl)
Foot	*Footnote counter (.ft, .fr)
FootRef	*Footnote reference string to be inserted in footnote body [string]
Fp	*First page to print (reset each pass to "From")
Fr	*Footnote counter reset switch (.fr)
From	*Argument of "from" option

Built-in Symbols

Symbol	Value
Hyphenating	*Hyphenation switch (hyphenate option)
In	Amount to indent (.in, .lm)
InputFileName	Name of current input file [string] (.if)
InputLines	Line number in current input file
LinesLeft	Number of usable text lines left on this page
Ll	*Line length (.ll, .rm)
Lp	*Last page to print (reset each pass to "To")
Ma1	*Space above header (.ma, .m1)
Ma2	*Space below header (.m2)
Ma3	*Space above footer (.m3)
Ma4	*Space below footer (.ma, .m4)
Ms	*Spacing between lines (.ms, .ss, .ds)
MultiplePagecount	*Number of form feeds between output pages (.mp)
NestingDepth	Index into stack of input files (.if)
Nl	Number of last used output line
NNp	*Next page number (.pa, start option)
NoFtNo	*Don't number next footnote (.fr)
NoPaging	*Suppress page breaks (paginate option)
Np	Current page number (.pa, start option, reset each pass from "Start")
OddFlagCol	*Column in which to flag odd pages (.fl)
OverprintCount	*Number of addtional times characters should be overprinted
PadLeft	*Pad from left end of line (.ad, complemented at end of each output line, set false at each break)
Parameter	*Passed argument betwen files (.if, paramter option)
Passes	*Number of passes left to make (=1 when printing is being performed) (passes option)
Pl	*Page length (.pl)
Print	*Print output ((Fp $\leq$ Np $\leq$ Lp) & (Passes $\leq$ 1))
PrintLineNumbers	*Source line numbers are being printed (number option)
Roman	*Number pages with Roman numerals (.ro/.ar)
Start	*Initial page number (start option)
Stopsw	*Pause between output pages (pause, stop and wait options)
TabStops	*Locations of tab stops [table] (.tb)
TextRef	*Footnote reference string inserted in main text [string]
Time	TENEX internal format of Date
To	*Last page to be printed (to option)
TrTable	*Translation table for output substitutions [table] (.tr)
Un	*Number of positions to decrease indenting (.un)

Built-in Symbols

Symbol	Value
UnderChar	*Character which is controlling underscoring (.uc)
Underscoring	*All characters are to be underscored (.uc, occurrences of double "UnderChar"s complement this)
Waitsw	*Wait before beginning output (wait option)
Widths	*Print width of output characters [table]

Hyphenation

By use of the "hyphenate" option, the user may request that the hyphenation routines attempt to break a word whenever the space available on an output line is less than the length of the next word (including attached punctuation, if any).

The hyphenation routines will split an already hyphenated word only at its hyphens. Beyond that, they do no syntactic analysis. They will interrogate the user for permissible hyphenations of every word that is a candidate to be split. When the user indicates the permissible hyphenation locations within a word, the word together with its hyphenations is entered into a dictionary so that if multiple passes are being done or if the word happens to be eligible again, the user will not be re-interrogated. The lookup in the dictionary for a match is a full string compare.

Before processing any of the input file, the user will be asked if he would like to pre-load the hyphenator's dictionary with a saved dictionary. If he does, the requested dictionary will be loaded and the user will not be interrogated about any words appearing in it. The format of a saved dictionary is an ASCII file containing words with embedded hyphens to indicate permissible hyphenation points. Tabs, carriage returns or form-feeds can be used to separate the words in the dictionary. All words should be entirely upper-case.

When during the course of processing the input file the user is interrogated for a hyphenation, the word in question will be typed out and the user should type in the same word, with hyphens to indicate where the word may be divided, or the user may type just a "." to indicate that the word should not be split. The user may edit his hyphenation with A, R or Q, and should terminate his input with a carriage return. Be careful that hyphenations are typed in upper-case; the hyphenator will always interrogate with words in that form.

At the end of each pass, if there have been any additions to the dictionary the user will be asked if he would like the accumulated dictionary saved. If so, the entire contents of the hyphenator's dictionary will be written out in a format suitable for later reloading.

If the user would like the capability of having words hyphenated as per a pre-loaded dictionary and at internal hyphens, but would rather not be interrogated about additional hyphenations, he should set AskHyphenations false.

MSG MANUAL

MSG was written by John Vittal for USC Information Sciences Institute.

MSG is a program for reading, writing, and subsectioning files which have a "message file format". It is very simple and straightforward to use. Commands are initiated by typing one character, which causes the program to type out the rest of the command name and wait for input from you.

Before the commands are described, there are a few general statements about how MSG works and some conventions used in describing the commands that you should know about. The prompt characters letting you know that MSG is waiting for a command character to be typed are "<-". When MSG is started up (by typing MSG<return> to the EXEC) it will first try to read your MESSAGE.TXT;1 file in your directory. If this file does not exist MSG will say so. If you were not connected to your login directory, MSG will try to find a MESSAGE.TXT;1 there. If that also fails, it will say so and wait for a command to be typed. If you have a MESSAGE.TXT;1, it will scan it and type out the header information (i.e. the date, from, and subject fields) for each message since the file was last read, preceded by a message number sequentially assigned by MSG. These message numbers are used in association with the various commands.

However, if you started MSG by typing MSG<space> to the EXEC, it will ask you for a file to be read. Typing an escape as the first character will cause MESSAGE.TXT;1 to be typed out, and confirmation requested from the user to ensure that that was what was intended. Once a file name has been specified and positively acknowledged, then the same information as described in the previous paragraph will be output to your terminal.

When reading a message file in MSG, either when starting up MSG or with the Read command described below, the file must be in the so-called "message file format". If MSG recognizes that the file does NOT conform to this format, you will be told so. However, you will be given the opportunity to keep everything that has been read so far, but NOT overwrite the 'bad' file. These two exceptional circumstances and some suggestions for getting around them are described at the end of this manual.

The following conventions and symbols are used in the command descriptions below. There are only five types of input MSG expects:

(1) a MSG command (or sub-command) character
(2) a message sequence specification
(3) a TENEX file name
(4) a confirmation character

(5) a local user name or remote site name

To abort output to the terminal type O (control-O). If MSG does not understand your input, it will return to command input mode, or reprompt you. The following are symbols and their associated meanings used in the command descriptions:

<FILE-NAME>

Stands for any TENEX file descriptor, including TTY: or LPT:. If you are requested to input a file name, the appropriate TENEX confirm will be given (e.g. [Old version]).

<MSG-SEQUENCE>

This input is prompted by the string (message sequence) in verbose typeout mode. A sequence of message numbers has the following format.

(1) Any single message number.

(2) Any two numbers separated by ">" or ":". This means message numbers delimited by the two outside numbers (e.g. 2>5 means messages 2,3,4,and 5 in that order). NOTE: if the first number is greater than the second number, it means the sequence in reverse order (e.g. 5>2 means messages 5,4,3, and 2).

(3) A pair of numbers separated by "-". This is so that the standard interpretation of the string "21-4" (that is not "21-24") means message numbers 21, 22, 23, and 24. Using this interpretation, the string and "24-1" is an error.

(4) Any sequence of the previous three types separated by commas. This is the way to group several non-adjacent messages together. For example: 1,3,5:7,1Ø means messages 1 and 3 and 5 through 7 and 1Ø.

<MSG-SEQUENCE> of the types described above are ALWAYS terminated by <return>.

(5) However, there are special types of message sequences. All are determined by the first character that you type in the <MSG-SEQUENCE> stream. The following are the twelve possibilities:

a. <escape> is typed, which causes the current message number to be echoed to you and the relevant process performed on that message only.

b. <control-I> is typed, which causes the previous completely specified <MSG-SEQUENCE> to be echoed and processing performed on that message stream.

c. R which stands for "Recent messages" only.

d. O standing for "Old messages" only.

e. A standing for "All messages" and which is equivalent to 1:(last message number).

f. D standing for "Deleted messages". This is valid ONLY in the context of the Headers, Undelete, and Delete commands. Everywhere else, the headers of the deleted messages will be printed. Of course, you can delete the typeout of those headers by typing control-O.

g. U standing for "Undeleted messages".
h. I standing for messages in inverse order. This is the opposite of the A (for all messages) sub-command.
i. S for "Subject field search for string" which asks you to provide a string which will be used as a mask match on the subject field of the message headers.
j. F for "From field search for string" which is like S but searches the Author field of the message headers instead. NOTE: the header command prints the initial part of the To: line of the message (if it exists) is the message was sent by the login-directory. Therefore, to search for messages sent by yourself, specify the string "To:" rather than the login directory name.
k. E standing for "Examined messages", i.e. all messages which have been completely typed (with the T command) or listed (with the L command).
l. N standing for "Not examined messages", which is the opposite of the E sub-command.
m. L standing for the "Last message sequence" that was completely specified.

Types (i) and (j) require you to type a string terminated by <return>. Typing just a <return> (i.e. the null string) means that searching is not to be performed. Otherwise, the search will be performed on the string typed up to (but not including) the <return>. The string you type must be an exact match to some substring of the appropriate field, but all alphabetic characters are treated as being upper case. (Note: carriage-retuns in the subject field of the header listing are ignored.)

(6) (Note: This is an experimental feature which may change in the future.) If you type comma or "M" as the first character of the message sequence that you are specifying, you will be able to specify more than one of the options drawn from the first five items mentioned here. You will then be entered into a sub-command mode. Any of the standard message sequences are acceptable as input. To terminate the specification of the list of message sequences, just type a carriage return in response to the prompt. If you wish to abort the acquisition at any time, type "Q" (for Quit) or control-N (N). To abort the acquisition of a single message sequence (like 3:14), type rubout. Typing rubout at the sub-command level (i.e. at the prompt without typing anything first) will have the same effect as typing control-N.

The default message sequence is 'All messages'. Any message sequence specified causes an intersection to be taken between that single message sequence (like 'Examined'), and the previous total. For example, the sequence:

```
<- Headers ,
<<- Examined
<<- From string: SYSTEM
<<-
```

would cause only the headers corresponding to messages from SYSTEM which have already been typed to get listed on your terminal.

If you just want to add a message sequence to the list, preface the actual message sequence with a "P" (for Plus) or "+". If you want to just subtract a message sequence from the list, preface the actual message sequence with an "M" (for Minus) or "-". For example,

```
<- Headers Multiple message sequences
<<- Examined
<<- From string: SYSTEM
<<- Plus: Subject string: MSG
<<- Minus: Deleted
<<-
```

will list the headers for all undeleted messages about MSG or which are examined messages from SYSTEM. No further associations between msg-sequence specifications are currently allowed.

In the command format below, everything that the program types will be lower case and everything you type will be in UPPER CASE. This is not the case when using MSG, but is used here for clarity.

MSG COMMANDS

<- Headers (message sequence) <MSG-SEQUENCE>

The headers for messages will be typed out for those messages defined by the message sequence typed. Headers corresponding to deleted messages have an asterisk printed before the header for that particular message. The headers for recent messages are preceded by a plus sign (+); messages which have not yet been typed are preceded by a minus sign (-), and deleted messages are preceded by an asterisk (*). If the message was sent by the user of the LOGIN directory, the initial part of the To: field of the message will be printed in the author field of the header, if the To: field exists in the message. In order to get the length of the message typed out along with the header, use the I command (which stands for Inclusion of length in header).

<- Delete (message sequence) <MSG-SEQUENCE>
This command will indicate (by a preceding asterisk) in the header information for the messages specified by <MSG-SEQUENCE> that those message are deleted. NOTE: This command marks each message in the actual message file indicating that it is deleted. If you reread the file for some reason, the messages will still be marked (and treated) as deleted (but not expunged). This command does however effect message numbers specified in later commands in the following way. If you have deleted message number 5 and then try to "Type" or "Put" message number 5 either directly or implied by the use of the ":" option, the deleted messages will NOT be included.

<- Undelete (message sequence) <MSG-SEQUENCE>
Of course! If you can delete a message, you certainly ought to be able to undelete it. This command undoes the action of the Delete command for the messages specified by this <MSG-SEQUENCE>.

Commands to See and Move Messages

<- Type (message sequence) <MSG-SEQUENCE>
This command will type on your terminal the messages specified by <MSG-SEQUENCE>. All messages which are completely typed are treated as having been 'examined'.

<- Put (message sequence) <MSG-SEQUENCE>
into file name: <FILE-NAME>
This command will put the messages specified by <MSG-SEQUENCE> into the file specified by <FILE-NAME>. If the file does not exist, it will create that file and write the messages into it. If the file already exists, it will append the messages to the messages already in the file. This command is useful if you want to keep separate files containing messages concerning different topics.

<- Move (message sequence) <MSG-SEQUENCE>
into file name: <FILE-NAME>
This command is a convenient combination of the Put and Delete commands. As the messages are put into the file, they are marked for deletion. If any of the messages are already deleted, you will be informed, and those messages will NOT be moved to the file.

<- List (message sequence) <MSG-SEQUENCE>
on file: <FILE-NAME>
This command lists all the specified messages on the file

specified. All messages specified by the <MSG-SEQUENCE> are treated as having been examined (typed). If you are listing more than one message, a preface page on the file will be created which contains the headers for those messages. You will be asked if you want each message on a separate page. This command is intended to allow a user to obtain a reasonable hard copy listing of some messages. (Note: the preface page of headers might have the length of each message included depending on the setting by the I(nclusion of length ih header) command.)

Commands to Update Your Message Files

<- Overwrite old file <FILE-NAME> [confirm]
This command will overwrite the current file (specified by <FILE-NAME>), reflecting the fact that you have deleted messages. That is, if you delete message 2 and then "overwrite" your file, message 2 will disappear from that file. It also rereads your file, renumbering your messages. You are warned if any unexamined messages (which are also not deleted) exist in the file that you are overwriting.

<- Quit [confirm]
This command returns you to the TENEX EXEC without rewriting any file (almost equivalent to typing control-C). You are warned if any unexamined messages (which are also not deleted) exist in the current message file.

<- Exit and update old file <FILE-NAME> [confirm]
This command is another way to Overwrite your old message file, but instead of rereading the file it returns you to the TENEX EXEC. This is equivalent to doing an overwrite followed by a Quit, but without the overhead of rereading the file. You are warned if any unexamined messages (which are also not deleted) exist in the file that you are overwriting.

<- Write file <FILE-NAME> sorted by message arrival time
This is similar in nature to the Overwrite command, except that the messages are sorted into ascending sequence by their arrival time before the overwriting is attempted. The file is then rescanned. You are warned if any unexamined messages (which are also not deleted) exist in the file that you are sorting.

Commands to Read Other Message Files

<- Read file name: <FILE-NAME>

You can use MSG on any file which has a "message format." This means you can peruse or modify files created with the "Put" or "Move" commands (but NOT the "List" command). If, for example, you have a file containing messages pertaining to MSG problems, you can read it to make sure you've taken care of them. Read is the command which lets you read files other than your standard mailbox, file MESSAGE.TXT;1. It also prints out the recent header information for that file. If that file has old messages which have not yet been 'examined', you will be informed. You will also be told if any of the old messages in the file are deleted.

Commands to Sequence through the Messages

<- Current message is nn of mm messages.
in file: <FILE-NAME>

This command tells you (1) the number of the current message, (2) the total number of messages, and (3) the file name of the currently active file. The current message is either the last message typed on your terminal or, if you have not typed one yet, either after the last message if the file had no recent messages, or before the first recent message. This command will let you know where the Next and Backing up commands will start, i.e. the first message they will type if used. Finally, it will tell you what the currently active message file is.

<- Go to message number: <NUMBER>

This will allow you to change the Current message number explicitly. If <NUMBER> is not in the range of acceptable numbers (i.e. it is less than 1 or greater than the number of messages in the file), or you did not type a number, you will be told and the Current message number will not be changed. However, there are several other options which are specified by the FIRST character typed:

a. E for the end of messages (the last message)
b. L for the last message (same as E).
c. B for the beginning of messages (message number 1)
d. escape (alt-mode) for current message number

<- Next message is:

This command types the message following the current message (if one exists) and sets the "current message" to be that message. Deleted messages will not be typed, but the "current message

number" will still be incremented.

<- <line feed>
Same as Next. Types the message following the current message, and sets the current message to be that message.

<- Backing up -- previous message is:
This command always types the previous message (i.e., Current message number - 1). It is the inverse of the Next command. It always decrements Current message number.

<-
This is equivalent to the Back command. It types the previous message and sets the current message to be that message.

<- H
The <control-H> (or New-line) command is equivalent to the Back command. It types the previous message and sets the current message to be that message.

Other commands

<- Verbose
This is a binary switch which causes the program to go into either 'Short typeout mode' or 'Long typeout mode', and tells you which is the setting that it changes to. The default is 'Short typeout mode'. Long typeout mode gives additional prompting regarding what is expected to be typed in.

<- Koncise
This is a binary switch which causes the program to go into either 'Concise typeout mode' or 'Short typeout mode' (the default), and tells you which is the setting that it changes to. Concise typeout mode shortens some of the typeout that MSG gives when it is interacting with the user. It is meant for 'advanced' users only.

<- Inclusion of length in header
This command is a binary switch which causes the program to go into a mode where header listings caused by the Header command will have the number of characters in the message included as part of the subject field. The default is that the length will

not be included. Note that when you read a file initially, the length of 'recent' messages will always be included in the initial listing of recent headers.

<- Zap profile [Confirm]

The Zap profile command will allow you to set up a user profile file for yourself without having to know the format of such a file. For the time being, the profile information will be limited. Typing control-N will exit you to the command level of MSG. Typing E at any point will 'Exit' the dialogue and ask you if you want the changes made permanently. At any point, type "?" to determine the appropriate responses. The following is a summary of the questions posed:

1. Normal, Verbose or Koncise typeout mode?
2. Always include the length of messages in all headers listings?
3. When in SNDMSG (from any of the Sndmsg, Forward or Answer commands), when you type control-N, do you want to abort Sndmsg without confirmation?
4. Do you want to be required to confirm all commands with a single carriage return?
5. Do you want to be told that some messages have been 'not-examined' whenever you try to quit MSG (by any of the Quit, Overwrite, or Exit and Update commands)?
6. Do you want to receive copies of your 'answers' to messages?
7. If the answer to (6) is yes, then you will be asked if you want to save all your 'answers' on the file SAVED.MESSAGES.
8. Do you want a list of headers for all messages:
   a. being deleted with either the "Move" or "Delete" commands
   b. being moved with the "Put" command
   c. being listed with the "List" or "Xerox" command
   d. being marked or unmarked with the "'" or "-" commands.

At the end of the dialogue, you will be asked if you want these changes in mode settings to be made permanent. If you answer 'Y' then each time you start up MSG, the settings of the modes noted here will be set to the values you indicated. Otherwise, the settings are set only for this session. They are NOT permanent, and can be changed any time.

<- : (prints current time and date)

<- ' Mark messages as examined (message sequence) <MSG-SEQUENCE>

This command will mark all the messages specified by <MSG-SEQUENCE> to be "examined", so MSG will think they have been typed or listed even though they may not have been.

<- - Unmark messages to be Not examined (message sequence)
<MSG-SEQUENCE>

This command will mark all the messages specified by <MSG-SEQUENCE> to be "NOT examined", so MSG will think they have NOT been typed or listed, even though they might have already been seen.

<- ; <COMMENT>

This command is mainly intended to allow you to talk with somebody over a link while you are in MSG. It eats all characters except <return>, control-Z (Z) and control-N (N), which return you to the command level of MSG. Two other characters have special effects. <delete> (<rub-out>) will type the string ' XXX ' and is useful in indicating that the previous word (or phrase) should be ignored. <line-feed> will cause effectively a carriage return and tab sequence to be typed. This way you can type more than one line of text. NOTE: the standard TENEX editing characters (e.g. control-A) are treated as any other character and perform no special function.

Command to Run Other Programs

<- Sndmsg [confirm]

This command will start up SNDMSG and give control of the terminal to it. When SNDMSG is finished (i.e. when you have sent the message), it will turn control back to MSG in the same state as it was before you sent the message. Control-N (N) will ask if you wish to abort. If you provide a positive confirmation, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

<- Answer message number: <MESSAGE-NUMBER>
Reply to those whom the message is: <ANSWER SUB-COMMAND>

This facility allows you to send a message to the sender of a message, and (at your discretion) those people to whom that message was sent, without having to type their addresses to Sndmsg.

The <ANSWER SUB-COMMAND> can be any of the following:

F -- From (indicating the sender of the message only)
T -- To (indicating the sender of the message and those addresses in the To: list)
C -- Cc (indicating all recipients of the original message in addition to the sender in the message)

Typing anything else aborts the command.

The <MESSAGE-NUMBER> can be any argument that the Go command takes:

a. a message number
b. E for the end of messages
c. L for the last message (same as E).
d. B for the beginning of messages (message number 1)
e. escape (alt-mode) for current message number
f. <return> for current message number.

The header of the message specified is also typed so that you may be sure you are answering the correct message. In fact, the header is typed after you have specified the message number, but before you are asked to supply the sub-command.

When prompted for additional addresses, any that you specify will be passed to SNDMSG as part of the cc: list. Some of the SNDMSG conventions are NOT implemented. These are the control-B feature which allows specification of a file, and the feature which allows you to specify a global host name (which spreads across several user names). Also, control-N aborts the Answer command! Local user names and remote host names are checked for validity.

An attempt is made to insure that all addresses are valid (i.e. all host names on remote addresses, and user names on local addresses), and that no duplications are present. If clarification is necessary from the user, you may be asked some questions. If these questions are posed, all type-ahead is deleted. If relevant, MSG will issue a warning if either the To: or cc: destination fields of the message have a destination list as part of the field (like LISP-USERS:). When control is given to you to type your answer, you will be typing to the message acquisition portion of SNDMSG (i.e. that part which normally would prompt you by typing "Message (? for help):"). Control-N (N) will ask if you wish to abort. If you give a positive confirmation, then you will be returned to the top level <.> MSG. Otherwise, you will be returned to SNDMSG.

There are two relevant profile mode settings. One is whether you wish to receive copies of the answers you send. It is initially assumed that you do. If you do not want copies of the replys you send, then your name will not appear in any of the destination lists unless you specify it as part of the additional carbon-copy list. However, if you do want copies of your messages, you will always receive one. In addition, if you have also indicated in your profile that you want all your responses to go to a file called SAVED.MESSAGES, then if that file exists, your responses will go in that file and NOT into your MESSAGE.TXT.

However, if you do not always want your answers to go to SAVED.MESSAGES, but do want copies of your answers, if there is a file named SAVED.MESSAGES in the login directory, you will be asked if you want your copy of the message to go to that file. If a positive response is given, then the login directory name will NOT appear in the destination lists.

<- Forward (message sequence) <MSG-SEQUENCE>
This facility will allow you to send copies of messages you have received to other people. The headers of the messages being forwarded are typed, after which you will be asked to provide the subject of this forwarded message. Then it will hand SNDMSG the subject and those messages you want forwarded, and leave you in SNDMSG in such a way that the message being forwarded can be edited, or your own comments added. You will be left in SNDMSG as though you had typed the forwarded message in yourself. When done, type a control-Z and then specify, in the standard way, to whom the mail is going. Once in SNDMSG, typing control-N (N) will ask if you wish to abort. If you give a positive confirmation in the standard way, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

<- Jump into lower fork running: <FILE-NAME>
This command is an escape in MSG in case you wish to run another program such as TECO, PUB, the EXEC, and so on. It searches directories to try to find the program you are asking it to run. The search list is, in order, <SUBSYS>, <SYSTEM>, your connected directory, and the login directory (if it is different from your connected directory). This way, you can run EXEC without having to type the complete information (i.e. <SYSTEM>EXEC.SAV).

If you decide to leave the lower fork, but want to continue it at a later time, all you need do is type an escape as the first character of the file name you are requested to provide. This will cause the old file name (preceded by an appropriate message) to be printed, and then you will be asked to confirm in the standard way. If you provide a positive confirmation, you will be asked if you want to continue or start that program. Typing 'C' for continue will put you back in the lower fork at the place where you exited; typing 'S' for start will restart the program.

<- Xed (editor) [confirm]
This command will start up XED (a text editor written at ISI). It has the capability to give SNDMSG the text built while in the editor as the body of the message. When you "Quit" XED you will return to MSG. Each additional time that you execute the XED command, you will be returned to the SAME copy of XED that you previously left with the XED "Quit" command (the old text buffers

are left intact).

<- Exec [confirm]
When you type control-E, the program will type "Exec" to you and ask for confirmation. This command is intended to give you a new copy of the EXEC with a minimum of hassles To leave that EXEC and return to MSG, type "QUIT". If you decide that you want a copy of the EXEC again, and you use this command, you will be given the same EXEC with all of your context intact.

This completes the list of MSG commands. There is only one item left to mention.

Receiving New Messages While Using MSG

MSG, on typing a command or returning from the execution of a command, checks to see if your currently active message file usually MESSAGE.TXT;1, has been written into. If it has, it prints out that fact and the headers for the new messages. The "current message number" is not modified. It then executes your command or returns to command mode, accordingly

Command Summary

Cmnd. Char. Meaning

A Answer message number: <MESSAGE-NUMBER>
Reply to whom the message is:
F - From
<return> -- same as F
T - To list plus original sender
C -- Cc list plus To: list plus original sender

B Backing up - previous message is
Same as Backing up

H Same as Backing up

C Current message is nn of mm messages
in file: <FILE-NAME>

D Delete (message sequence) <MSG-SEQUENCE>

E Exec [confirm]

E Exit and update old file <FILE-NAME> [confirm]

F Forward (message sequence) <MSG-SEQUENCE>

G Go to message number: <MESSAGE-NUMBER>

H Headers (message sequence) <MSG-SEQUENCE>

I Inclusion of length in header

J Jump into lower fork running file: <program name> [confirm]

K Koncise -- provides shorter prompting

```
    L   List (message sequence) <MSG-SEQUENCE>
        on file name: <FILE-NAME>
    M   Move (message sequence) <MSG-SEQUENCE>
        into file name: <FILE-NAME>
    N   Next message is:
  <lf>          (line feed) same as Next message is:
    O   Overwrite old file <FILE-NAME> [confirm]
    P   Put (message sequence) <MSG-SEQUENCE>
        into file name: <FILE-NAME>
    Q   Quit [confirm]
    R   Read file name:  <FILE-NAME>
    S   Sndmsg [confirm]
    T   Type (message sequence) <MSG-SEQUENCE>
    U   Undelete (message sequence) <MSG-SEQUENCE>
    V   Verbose -- provides more prompting
    W   Write file <FILE-NAME> sorted by message arrival time [confirm]
    X   Xed  [confirm]
    Z   Zap profile [confirm]
    '   '   Mark messages as 'examined' (message sequence) <MSG-SEQUENCE>
    -   -   Unmark messages to be NOT 'examined' (message sequence)
                                                 <MSG-SEQUENCE>
    :   :   (the time and date is then printed)
    ?   ?   Type command character for its description, ? for summary
    ;   ;   Comment -- <return> or Z returns you to command level
```

Abort commands on typein and terminal output with control-N (N).
Confirm with Y or <return>.

Errors While Reading a Message File

When reading a file in MSG (either at startup or with the 'Read' command), the file MUST be in the so-called "message file format". If MSG recognizes that the file does NOT conform to this format, you will be told so. The following are the circumstances which might cause the file to become unreadable, and some suggestions for getting around the problems.

The file is a message file (that is, one or more valid messages have been read from it), but somewhere in the middle it does not conform to the message file format. It could be: (1) It has a hole in it. Read the file with a text editor to get rid of the hole, and write it back out, and reuse MSG. Try this first. If this doesn't work, MSG will give you an error at the same place. Then you can try the second suggestion: (2) If suggestion 1 didn't work, then the file has internal byte counts which do not match the actual file. Either you used a text editor on your message file changing the number of bytes but not the byte counts or your file was mysteriously altered. The date of a message could not be read. Either the byte count for the last message read was wrong, or there is junk between the last message read