5. MM MEMORANDUM MACROS USER GUIDE

A. Introduction

A.1 Purpose

This chapter is a guide and reference manual for users of Memorandum Macros (MM). These macros provide a general purpose package of text formatting macros for use with the typesetting programs

Each part of this chapter explains a single facility of MM and progresses from general case to special-case facilities. It is recommended that a user read a part in detail only to the point where there is enough information to obtain the desired format, then skim the rest because some details may be of use to only a few.

A.2 Conventions

Items enclosed in braces (**{}**) refer to the section and paragraph numbers within the section. For example, this is paragraph {§A.2}.

In the synopses of macro calls, square brackets (**[ ]**) surrounding an argument indicate that it is optional. Ellipses (**...**) show that the preceding argument may appear more than once.

In those cases in which the behavior of the two formatters *nroff* and *eroff/troff* is obviously different, the *nroff* formatter output is described first with the *eroff/troff* formatter output following in parentheses.

For Example:

> The title is underlined (Italic).

means that the title is underlined by the *nroff* formatter and italicized by the *eroff* and *troff* formatters.

A.3 Input Text Structure

In order to make it easy to edit or revise input file text at a later time the following style is recommended:

- Keep input lines short.
- Break lines at the end of clauses.
- Begin each new sentence on a new line.

A.4 Definitions

Formatter refers to either the *nroff, eroff,* or *troff* text formatting programs. The *eroff* program is an enhanced interface to *troff*. For purposes of studying this document, you may consider *eroff* and *troff* to be interchangeable terms.

Requests are built-in or *low-level* commands. One common request is

```
.sp
```

which generates a blank line. A user seldom needs to use these requests directly; instead the commands discussed in this chapter are often preferred. (In this case, the **mm** macro **P** is

usually used instead of **sp**.

Macros are named collections of requests. Each macro is an abbreviation for a collection of requests that would otherwise need to be repeated at various points in a document. The *mm* package supplies many macros, and the user can define additional ones. Macros and requests share the same set of names and are used in the same way.

A complete listing of memorandum macros is given in the *MM Macro Name Summary* {§R} of this chapter.

Strings provide character variables, each of which names a string of characters. Strings are often used in page headers, page footers, and lists. A string can be given a value via the **ds** (define string) request, and its value can be obtained by referencing its name, preceded by "`\*`" (for 1-character names) or "`\*(`" (for 2-character names). For instance, the string **DT** in MM normally contains the current date, thus the input line

```
Today is \*(DT.
```

may result in the following output:

Today is July 25, 1992.

The current date can be replaced (in any format), e.g.:

```
.ds DT 07/25/92
```

by invoking a macro designed for that purpose {§F.8}. A listing of MM string names is given in the *MM String Name Summary* {§T} of this chapter.

Strings share the pool of names used by requests and macros.

Number registers fill the role of integer variables. These*r egisters* are used for flags and for arithmetic and automatic numbering. A register can be given a value using an **nr** request and be referenced by preceding its name by "`\n`" (for 1-character names) or "`\n(`" (for 2-character names). For example, the following sets the value of the register **d** to one more than that of the register **dd**:

```
.nr  d  1+\n(dd
```

A complete listing of MM number registers is given in the *MM Number Register Summary* {§S} section of this chapter.

The final sections of this chapter list all macros, strings, number registers, and error messages defined in MM.

B. Usage

This section describes how to access MM.

B.1 The −mm Flag

The MM package can also be included by including the `-mm` flag as an argument to the formatter. The`-mm` flag causes the file /usr/lib/tmac/tmac.m to be read and processed before an y other files. This action:

- defines the Memorandum Macros
- sets default values for various parameters

- initializes the formatter to be ready to process input text files.

B.2 Typical Command Lines

The prototype command lines are as follows (various options are explained in paragraph B.3):

- Text without tables, equations, or pic diagrams:

  eroff [options] −mm file . . .
  nroff [options] −mm file . . .
  troff [options] −mm file . . .

- Text with tables:

  tbl -D file . . . | eroff [options] −mm
  tbl -D file . . . | troff [options] −mm
  tbl -D file . . . | nroff [options] −mm

- Text with equations:

  eqn /usr/pub/eqnchar file . . . | eroff [options] −mm
  eqn /usr/pub/eqnchar file . . . | troff [options] −mm
  neqn /usr/pub/eqnchar file . . . | nroff [options] −mm

- Text with pic diagrams:

  pic -D file . . . | eroff [options] −mm
  pic -D file . . . | nroff [options] −mm
  pic -D file . . . | troff [options] −mm

Files may contain both tables and equations, or pic diagrams, or all three. In this case, just include the necessary processor in the command *pipe-line*. For example, to format a document with all three, you would say:

pic -D file | eqn | tbl -D | eroff [options] −mm

When formatting a document, the output would normally be processed for a specific type of printer or terminal. The formatter option `-T` is used to specify the type of output device you intend to use. With *nroff*, the default output device is the Model-33 Teletype. With *eroff* (and *troff*), the default output device is the APS typesetter. A list of valid printer/terminal types is given in the **REFERENCE** section under *nroff* and *eroff*.

If 2-column processing {§M.4} or table processing is used with the *nroff* formatter, the output may need to be postprocessed by **col**.

B.3 Parameters Set From Command Line

Number registers are commonly used within MM to hold parameter values that control various aspects of output style. Many of these values can be changed within the text files with **nr** requests. In addition, some of these registers can be set from the command line. This is a useful feature for those parameters that should not be permanently embedded within the input text. If used, the number registers (with the exception of the **P** register) must be set on the command line or before the MM macro definitions are processed. The number register meanings are:

`-rA`*n* *n* = 1 has effect of invoking the **AF** macro without an argument {§F.9}.

`-rC`*n* sets type of copy (e.g., DRAFT) to be printed at bottom of each page {§J.2.4}.
n = 1 for OFFICIAL FILE COPY.

n = 2 for DATE FILE COPY.
n = 3 for DRAFT with single spacing and default paragraph style.
n = 4 for DRAFT with double spacing and 10-space paragraph indent.

`-rD1` sets *debug* mode.
This flag requests formatter to continue processing even if MM detects errors that would otherwise cause termination. It also includes some debugging information in the default page header {§J.2.1,J.3}.

`-rE`*n* controls font of Subject/Date/From fields.
n = 0, fields are bold (default for the *troff* formatter).
n = 1, fields are Roman font (regular text-default for the *nroff* formatter).

`-rL`*k* sets length of physical page to *k* lines.
For the *nroff* formatter, *k* is an unscaled number representing lines.
For the *troff* formatter, *k* must be scaled. (e.g., `-rL8.5i`)
Default value is 66 lines per page (11i).

`-rN`*n* specifies page numbering style (See Figure 5-1).
n = 0 (default), all pages get the prevailing header {§J.2.1}.
n = 1, page header replaces footer on page 1 only.
n = 2, page header is omitted from page 1.
n = 3, ‘‘section-page’’ numbering {§D.5} occurs (.FD {§I.3} and .RP {§L.4} defines footnote and reference numbering in sections).
n = 4, default page header is suppressed; however, a user-specified header is not affected.
n = 5, ‘‘section-page’’ and ‘‘section-figure’’ numbering occurs.

n	*Page 1*	*Pages 2–End*
0	header	header
1	header replaces footer	header
2	no header	header
3	‘‘section-page’’ as footer	same as page 1
4	no header	no header unless .PH defined
5	‘‘section-page’’ as footer and ‘‘section-figure’’	same as page 1

Figure 1. Page/Figure Numbering Style (N register)

Contents of the prevailing header and footer do not depend on number register *N* value; *N* controls whether the header (*N*=3) or the footer (*N*=5) is printed, as well as the page numbering style. If header and footer are null {§J.2.1,J.2.4}, the value of *N* is irrelevant.

`-rO`*k* (register capital letter ‘‘O’’) offsets output *k* spaces to the right. For the *nroff* formatter, *k* is an unscaled number representing lines or character positions. For the *troff* formatter, *k* must be scaled. This flag is helpful for adjusting output positioning on some terminals. The default offset if this register is not set on the command line is 0.75 inch (*nroff*) and 0.5 inch (*troff*).

`-rP`*n* specifies that pages of the document are to be numbered starting with *n*. This register may also be set via an **nr** request in the input text.

`-rS`*n* sets point size and vertical spacing for the document. The default *n* is 10, i.e., 10-point type on 12-point vertical spacing, giving six lines per inch {§M.10}. This flag applies to the *troff* formatter only.

`-rT`*n* provides register settings for certain devices. If*n* is 1, line length and page of fset are set to 80 and 3, respectively. Setting*n* to 2 changes the page length to 84 lines per page and inhibits underlining. The default value for *n* is 0. This flag applies to the *nroff* formatter only.

`-rU1` controls underlining of section headings. This flag causes only letters and digits to be underlined. Otherwise, all characters (including spaces) are underlined {§D.2.2.4.2}. This flag applies to the*nr off* formatter only.

`-rW`*k* sets page width (line length and title length) to *k*. For the *nroff* formatter, *k* is an unscaled number representing character positions. For the *troff* formatter, *k* must be scaled. This flag can be used to change page width from the default value of 6 inches (60 characters in 10 pitch or 72 characters in 12 pitch).

B.4 Omission of −*mm* Flag

If a large number of arguments is required on the command line, it may be convenient to set up the first (or only) input file of a document as follows:

initialization of registers listed above
`.so /usr/lib/tmac/tmac.m`
remainder of text

In this case, the user should not use the `-mm` flag because the **so** request has the equivalent effect, but registers above must be initialized before the **so** request because their values are meaningful only if set before macro definitions are processed. When using this method, it is best to lock into the input file only those parameters that are seldom changed. For example:

```
.nr W 80n
.nr O 10n
.nr N 3
.so /usr/lib/tmac/tmac.m
.H 1 ‘‘Introduction’’
  .
  .
  .
```

specifies, for the *nroff* formatter, a line length (register **W**) of 80 characters, a page offset (register **O**) of 10 characters, and ‘‘section-page’’ (register **N**) numbering.

C. Formatting Concepts

C.1 Basic Terms

The normal action of the formatters is to fill output lines from one or more input lines. Output lines may be justified so that both the left and right margins are aligned. As lines are being filled, words may also be hyphenated {§C.4} as necessary. It is possible to turn any of these modes on and off (**SA** {§M.2}, **Hy** {§C.4}, and the **nf** and **fi** formatter requests). Turning off fill mode also turns off justification and hyphenation.

Certain formatting commands (requests and macros) cause filling of the current output line to cease, the line (of whatever length) to be printed, and subsequent text to begin a new output line. This printing of a partially filled output line is known as a *break*. A few formatter requests and most of the MM macros cause a *break*.

> ''The single most important rule in using *troff* is not to use it directly, but through some intermediary.''

Formatter requests {§C.10} can be used with MM; however, there are consequences and side effects that each such request might have. A good rule is to use formatter requests*only when absolutely necessary*. The MM macros described herein should be used in most cases because:

- It is much easier to control (and change at any later point in time) the overall style of the document.
- Complicated features (such as footnotes, tables of contents, etc.) can be obtained with ease.
- The user is insulated from peculiarities of the *low-level* formatter language.

C.2 Arguments and Double Quotes

For any macro call, a null argument is an argument whose width is zero. Such an argument often has a special meaning; the preferred form for a null argument is two double quotes: `''''` Omitting an argument is not the same as supplying a null argument (e.g., the **MT** macro {§F.7}). Omitted arguments can occur only at the end of an argument list; null arguments can occur anywhere in the list.

Any macro argument containing ordinary (paddable) spaces must be enclosed in double quotes; otherwise, it will be treated as several separate arguments. A double quote (`''`) is a single character that must not be confused with two apostrophes (`''`), or grave accents (`` `` ``).

Double quotes are not permitted as part of the value of a macro argument or of a string that is to be used as a macro argument. If it is necessary to have a macro argument value, two grave accents (`` `` ``) and/or two acute accents (´´) may be used instead. This restriction is necessary because many macro arguments are processed (interpreted) a variable number of times. For example, headings are first printed in the text and may be reprinted in the table of contents.

C.3 Unpaddable Spaces

When output lines are justified to give an even right margin, existing spaces in a line may have additional spaces appended to them. This may distort the desired alignment of text. To avoid this distortion, it is necessary to specify a space that cannot be expanded during justification, i.e., an *unpaddable space*. There are several ways to accomplish this.

- The user may type a backslash followed by a space (''`\ `''). This pair of characters directly generates an unpaddable space.
- The user may sacrifice some seldom-used character to be translated into a space upon output. The chosen character may be used anywhere an unpaddable space is desired. The tilde (~) is often used with the translation macro for this purpose. To cause the ~ character to be translated to a (unpaddable) space, the following may be inserted at the beginning of the document:

```
.tr ~
```

If a tilde must actually appear in the output, it can be temporarily ‘‘recovered’’ by inserting

```
.tr ~~
```

before the place where needed. Its previous usage is restored by repeating the `.tr ~` after a break or after the line containing the tilde has been forced out.

C.4 Hyphenation

Formatters do not perform hyphenation unless requested. Hyphenation can be turned *on* in the body of the text by specifying

```
.nr Hy 1
```

at the beginning of the document input file. See {§I.3} for a discussion of hyphenation within footnotes and across pages.)

If hyphenation is requested, the formatters will automatically hyphenate words if need be. However, the user may specify hyphenation points for a specific occurrence of any word with a special character known as a hyphenation indicator or may specify hyphenation points for a small list of words (about 128 characters).

If the *hyphenation indicator* (initially, the 2-character sequence ‘‘\%’’) appears at the beginning of a word, the word is not hyphenated. Alternatively, it can be used to indicate legal hyphenation points inside a word. All occurrences of the hyphenation indicator disappear on output.

The user may specify a different hyphenation indicator.

`.HC` [*hyphenation-indicator*]

The circumflex (^) is often used for this purpose by inserting the following at the beginning of a document input text file:

```
.HC ^
```

Any word containing hyphens or dashes will be hyphenated immediately after a hyphen or dash if it is necessary to hyphenate the word, even if the formatter hyphenation function is turned off.

The user may also supply, via the exception word **hw** request, a small list of words with the proper hyphenation points indicated. For example, to indicate the proper hyphenation of the word ‘‘printout’’, the user may specify

```
.hw print-out
```

C.5 Tabs

Macros **MT** {§F.7}, **TC** {§K.1}, and **CS** {§K.2} use the **ta** (tabs) formatter request to set tab stops and then restore the default values of tab settings (every eight characters in the *nroff* formatter; every ½ inch in the *troff* formatter). Setting tabs to other than the default values is the user’s responsibility.

Default tab setting values for *nroff* are 9, 17, 25, . . ., 161 for a total of 20 tab stops. Values may be separated by commas, spaces, or any other non-numeric characters. A user may set tab stops at any value desired. For example:

```
.ta 9 17 25 33 41 49 57 . . . 161
```

A tab character is interpreted with respect to its position on the input line rather than its position on the output line. In general, tab characters should appear only on lines processed in no-fill (**nf**) mode {§C.1}.

N.B.: The *tbl* program {§H.3} changes tab stops but does not restore default tab settings.

C.6 BEL Character

The nonprinting character BEL (Ctrl-G, ASCII code 7) is used as a delimiter in many macros to compute the width of an argument or to delimit arbitrary text, e.g., in page headers and footers {§J}, headings {§D}, and lists {§E}. Users who include BEL characters in their input text file (especially in arguments to macros) will receive mangled output.

C.7 Bullets

A bullet (•) is often obtained on a typewriter terminal by using an ‘‘o’’ overstruck by a ‘‘+’’. For compatibility between the *nroff* and *troff* formatters, a bullet string is provided by MM with the following sequence:

```
\*(BU
```

The bullet list (**BL**) macro {§E.1.1.2} uses this string to generate automatically the bullets for bullet listed items.

C.8 Dashes, Minus Signs, and Hyphens

The *troff* formatter provides distinct graphics for an em dash, an en dash, a minus sign, and a hyphen; the *nroff* formatter does not. The following approach is suggested:

Em Dash — Type ‘‘*(EM’’ for each em dash for both *nroff* and *troff* formatters. This string generates an em dash (‘‘—’’) in the *troff* formatter and two hyphens (`--`) in the *nroff* formatter. The dash list (**DL**) macro {§E.1.1.3} automatically generate the *em dash* for each list item.

En Dash — Type ‘‘*(En’’ for each en dash for both *nroff* and *troff* formatters. This string generates an en dash (‘‘–’’) in the *troff* formatter and a minus (`-`) in the *nroff* formatter.

Minus — Type ‘‘\-’’ for a minus sign. The *nroff* formatter will ignore the \. The*tr off* formatter will print a true minus sign (‘‘−’’).

Hyphen — Type ‘‘-’’ for a hyphen. The*nr off* formatter will print it as is. The *troff* formatter will print a true hyphen (‘‘-’’).

C.9 Trademark String

A trademark string ‘‘*(Tm’’ is available with MM. This places the trademark symbol (‘‘™’’) after the text that it follows.

C.10 Registered Trademark String

A registered trademark string ‘‘*(Rg’’ is also available with MM. This places the registered trademark symbol (‘‘®’’) after the text that it follows.

For example:

```
The
.I
UNIX
\*(Rg
.I
System User Reference Manual
.R
is available from the library.
```

yields:

The *UNIX® System User Reference Manual* is available from the library.

C.11 Use of Formatter Requests

Most formatter requests should not be used with MM because MM provides the corresponding formatting functions in a much more user-friendly and surprise-free fashion than do the basic formatter requests. However, some formatter requests are useful with MM, and we list them here:

`.af`	Assign format
`.br`	Break
`.ce`	Center
`.de`	Define macro
`.ds`	Define string
`.fi`	Fill output lines
`.hw`	Exception word
`.ls`	Line spacing
`.nf`	No filling of output lines
`.nr`	Define and set number register
`.nx`	Go to next file (does not return)
`.rm`	Remove macro
`.rr`	Remove register
`.rs`	Restore spacing
`.so`	Switch to source file and return
`.sp`	Space
`.ta`	Tab stop settings
`.ti`	Temporary indent
`.tl`	Title
`.tr`	Translate
`.!`	Escape

The **fp**, **lg**, and **ss** requests are also sometimes useful for the *troff* formatter. Use of other requests without fully understanding their implications very often leads to disaster! *Caveat Emptor!*

D. Paragraphs and Headings

D.1 Paragraphs

`.P` [*type*]
one or more lines of text.

The **P** macro designates a new paragraph follows. The*type* is used to control paragraph indention.

D.1.1 Paragraph Indention

An indented or a nonindented paragraph is defined with the *type* argument:

Type	*Result*
0	left-justified first line
1	indented first line

With type 0 the first line begins at the left margin. With type 1 the paragraph is indented.

The indention amount is controlled by the **Pi** register, which defaults to 5. For example, to indent paragraphs by ten spaces, the following is entered at the beginning of the document input file:

```
.nr Pi 10
```

A document input file possesses a default paragraph type obtained by specifying `.P` before each paragraph that does not follow a heading {§D.2}. The default paragraph *type* is controlled by the **Pt** number register:

- The initial value of **Pt** is 0, which provides left-justified paragraphs.
- *All* paragraphs can be forced to be indented by inserting the following at the beginning of the document input file:

  ```
  .nr Pt 1
  ```

- All paragraphs can be indented except after headings, lists, and displays by entering the following at the beginning of the document input file:

  ```
  .nr Pt 2
  ```

Both the **Pi** and **Pt** register values must be greater than zero for any paragraphs to be indented. Values that specify indention must be unscaled and are treated as character positions, i.e., as a number of ens. In the *nroff* formatter, an en is equal to the width of a character. In the*tr off* formatter, an en is the number of points (720 points = 1 inch) equal to half the current point size.

Regardless of the value of **Pt**, an individual paragraph can be forced to be left-justified or indented. The `.P 0` macro call forces left justification; `.P 1` causes indention by the amount specified by the register **Pi**.

If `.P` occurs inside a list, the indent (if any) of the paragraph is added to the current list indent {§E}.

D.1.2 Numbered Paragraphs

Numbered paragraphs may be produced by setting the **Np** register to 1. This produces paragraphs numbered within first level headings, e.g., 1.01, 1.02, 1.03, 2.01, etc:

5.01 This is an example of a paragraph that has been numbered by setting the number register **Np** to 1. The paragraph was preceded by a simple `.P` command.

Numbered paragraphs may also be obtained by using the **nP** macro rather than the **P** macro for paragraphs. This produces paragraphs that are numbered within second level headings.

```
.H 1 ''First Heading''
.H 1 ''Second Heading''
.nP
```
one or more lines of text

The format is slightly different in that the paragraphs contain a ''double-line indent''—The text of the second line is indented to be aligned with the text of the first line so that the number stands out. Here's an example:

1.01 This is an example of a paragraph that has been numbered via **nP** used in place of **P**. Notice that the first and second lines are indented to make the number stand out and the third and subsequent lines are not indented.

D.1.3 Spacing Between Paragraphs

The **Ps** number register controls the amount of spacing between paragraphs. By default, **Ps** is set to 1, yielding one blank space (one-half a vertical space).

D.2 Numbered Headings

`.H` *level* [*heading-text*] [*heading-suffix*]
zero or more lines of text

The *level* argument provides the numbered heading level. There are seven heading levels; level 1 is the highest, level 7 is the lowest.

The *heading-text* argument is the text of the heading. If the heading contains more than one word or contains spaces, the entire argument must be enclosed in double quotes.

The *heading-suffix* argument may be used for footnote marks which should not appear with heading text in the table of contents.

There is no need for a `.P` macro immediately after a `.H` or `.HU` {§D.3} because the **H** macro also performs the function of the **P** macro. Any immediately following **P** macro is ignored. It is, however, good practice to start every paragraph with a **P** macro, thereby ensuring that all paragraphs uniformly begin with a **P** throughout an entire document.

D.2.1 Normal Appearance

The effect of the **H** macro varies according to the *level* argument. First-level headings are preceded by two blank lines (one vertical space); all others are preceded by one blank line (one-half a vertical space). The following table describes the default effect of the *level* argument.

`.H 1` *heading-text*	Produces an underlined (Italicized) font heading followed by a single blank line (one-half a vertical space). The following text begins on a new line and is indented according to the current paragraph type. Full capital letters should be used to make the heading stand out.
`.H 2` *heading-text*	Produces an underlined (Italicized) font heading followed by a single blank line (one-half a vertical space). The following text begins on a new line and is indented according to the current paragraph type. Initial capitals should be used in the heading text.
`.H` *n heading-text*	$3 \le n \le 7$. Produces an underlined (Italicized) heading followed by two spaces The following text begins on the same line.

Appropriate numbering and spacing (horizontal and vertical) occur even if the heading-text argument is omitted from a `.H` macro call.

The following listing gives the first few `.H` calls used for this part:

```
.H 1 ‘‘Paragraphs and Headings’’
.H 1 ‘‘Paragraphs’’
.H 3 ‘‘Paragraph Indention’’
.H 3 ‘‘Numbered Paragraphs’’
.H 3 ‘‘Spacing Between Paragraphs’’
.H 1 ‘‘Numbered Headings’’
.H 3 ‘‘Normal Appearance’’
.H 3 ‘‘Altering Appearance’’
.H 4 ‘‘Prespacing and Page Ejection’’
.H 4 ‘‘Spacing After Headings’’
.H 4 ‘‘Centered Headings’’
.H 4 ‘‘Bold, Italic, and Underlined Headings’’
.H 5 ‘‘Control by Level’’ :
```

Users satisfied with the default appearance of headings may skip to the paragraph entitled ‘‘Unnumbered Headings’’ {§D.3}.

D.2.2 Altering Appearance

The user can modify the appearance of headings quite easily by setting certain registers and strings at the beginning of the document input text file. This permits quick alteration of a document’s style because this style-control information is concentrated in a few lines rather than being distributed throughout the document.

D.2.2.1 Prespacing and Page Ejection

A first-level heading (`.H 1`) normally has two blank lines (one vertical space) preceding it, and all other headings are preceded by one blank line (one-half a vertical space). If a multiline heading were to be split across pages, it is automatically moved to the top of the next page. Every first-level heading may be forced to the top of a new page by inserting:

```
.nr Ej 1
```

at the beginning of the document input text file. Long documents may be made more manageable if each section starts on a new page. Setting the **Ej** register to a higher value causes the same effect for headings up to that level, i.e., a page eject occurs if the heading level is less than or equal to the **Ej** value.

D.2.2.2 Spacing After Headings

Three registers control the appearance of text immediately following a .H call. The registers are **Hb** (heading break level), **Hs** (heading space level), and **Hi** (post-heading indent).

- If the heading level is less than or equal to **Hb**, a break {§C.1} occurs after the heading.
- If the heading level is less than or equal to **Hs**, a blank line (one-half a vertical space) is inserted after the heading.
- If a heading level is greater than **Hb** and also greater than **Hs**, then the heading (if any) is immediately followed by text on the same line.

These registers permit headings to be separated from the text in a consistent way throughout a document while allowing easy alteration of white space and heading emphasis. The default value for **Hb** and **Hs** is 2.

For any stand-alone heading, i.e., a heading on a line by itself, alignment of the next line of output is controlled by the **Hi** number register.

- If **Hi** is 0, text is left-justified.
- If **Hi** is 1 (the default value), text is indented according to the paragraph type as specified by the **Pt** register {§D.1.1}.
- If **Hi** is 2, text is indented to line up with the first word of the heading itself so that the heading number stands out more clearly.

To cause a blank line (one-half a vertical space) to appear after the first three heading levels, to have no run-in headings, and to force the text following all headings to be left-justified (regardless of the value of **Pt**), the following should appear at the beginning of the document input text file:

```
.nr Hs 3
.nr Hb 7
.nr Hi 0
```

D.2.2.3 Centered Headings

The **Hc** register can be used to obtain centered headings. A heading is centered if its *level* argument is less than or equal to **Hc** and if it is also a stand-alone heading {§D.2.2.2}. The **Hc** register is 0 initially (no centered headings).

D.2.2.4 Heading Font Control

D.2.2.4.1 Control by Level:

The string **HF** (heading font) contains seven codes that specify fonts for heading levels 1 through 7. Legal codes, code interpretations, and defaults for **HF** codes are shown in Figure 5-2.

HF Code	*Nroff Font*	*Troff Font*
1	Regular	Roman (R)
2	Underline	Italic (I)
3	Bold	Bold (B)
4	Bold Underline	Bold Italic (BI)
5		Helvetica (H)
6		Helvetica Oblique (HI)
7		Helvetica Bold (HB)
8		Helvetica Bold Oblique (HX)

Figure 2. HF String Codes, Effects, and Default Values

N.B.: Codes 4 and above are typesetter dependent. To determine your system's default fonts, run the command:

`troff -T`*typesetter* `< /dev/null` (On UNIX)

or

`troff -T`*typesetter* `< nul` (On MS-DOS)

For example:

```
troff -Tlj+.2 < /dev/null
```

The *troff* command will list the fonts and code numbers in the form:

`x font` *N Name*

where *N* is the code and *Name* is the font name.

All levels default to 2 (underlined in *nroff*, Italicized in *troff*). The user may reset**HF** as desired. Any value omitted from the right end of the list is assumed to be a 1.

The following request would result in one top level Helvetica Bold, four Roman Bold levels, and two Roman Italic levels:

```
.ds HF 7 3 3 3 3 2 2
```

D.2.2.4.1.1 NROFF Underlining Style: The *nroff* formatter underlines in either of two styles:

- The normal style (**ul** request) is to underline only letters and digits.
- The continuous style (**cu** request) underlines all characters, including spaces.

By default, MM attempts to use the continuous style on any heading that is to be underlined and is short enough to fit on a single line. If a heading is to be underlined but is longer than a single line, the heading is underlined in the normal style.

All underlining of headings can be forced to the normal style by using the `-rU1` flag when invoking the *nroff* formatter {§B.3}.

D.2.2.4.1.2 Heading Point Sizes: The user may specify the desired point size for each heading level with the **HP** string (for use with the *troff* formatter only).

`.ds HP` [*ps1*] [*ps2*] [*ps3*] [*ps4*] [*ps5*] [*ps6*] [*ps7*]

If **HP** is not set, the text of headings (`.H` and `.HU`) is printed in the same point size as the body except: *bold stand-alone headings are printed one point smaller than the body*.

The string **HP**, similar to the string **HF**, can be specified to contain up to seven values, corresponding to the seven levels of headings. For example:

```
.ds HP 12 12 10 10 10 10 10
```

specifies that the first and second level headings are to be printed in 12-point type with the remainder printed in 10-point. Specified values may also be relative point-size changes, for example:

```
.ds HP +2 +2 -1 -1
```

If absolute point sizes are specified, then absolute sizes will be used regardless of the point size of the body of the document. If relative point sizes are specified, then point sizes for headings will be relative to the point size of the body even if the latter is changed.

Null or zero values imply that default size will be used for the corresponding heading level. If an entry is non-null, the default rule for Bold point size reduction is *not* used.

Note: Only the point size of the headings is affected. Specifying a large point size without providing increased vertical spacing (via `.HX` and/or `.HZ` {§D.6}) may cause overprinting.

D.2.2.5 Marking Styles—Numerals and Concatenation

`.HM` [*arg1*] . . . [*arg7*]

Registers **H1** through **H7** are used as counters for the seven levels of headings. Register values are normally printed using Arabic numerals. The **HM** macro (heading mark style) allows this choice to be overridden, thus providing ‘‘outline’’ and other document styles. This macro can

have up to seven arguments; each argument is a string indicating the type of marking to be used. Legal arguments and their meanings are:

Argument	*Meaning*
1	Arabic (default for all levels)
0001	Arabic with enough leading zeroes to get the specified number of digits
A	Uppercase alphabetic
a	Lowercase alphabetic
I	Uppercase Roman
i	Lowercase Roman
omitted	Interpreted as 1 (Arabic)
illegal	No effect

By default, the complete heading mark for a given level is built by concatenating the mark for that level to the right of all marks for all levels of higher value. To inhibit the concatenation of heading level marks, i.e., to obtain just the current level mark followed by a period, the heading mark type register (**Ht**) is set to 1.

For example, a commonly used ‘‘outline’’ style is obtained by:

```
.HM I A 1 a i
.nr Ht 1
```

D.3 Unnumbered Headings

`.HU` *heading-text*

The **HU** macro is a special case of **H**; it is handled in the same way as **H** except that no heading mark is printed. In order to preserve the hierarchical structure of headings when `.H` and `.HU` calls are intermixed, each `.HU` heading is considered to exist at the level given by register **Hu**, whose initial value is 2. Thus, in the normal case, the only difference between:

`.HU` *heading-text*

and

`.H 1` *heading-text*

is the printing of the heading mark for the latter. Both macros have the effect of incrementing the numbering counter for level 2 and resetting to zero the counters for levels 3 through 7. Typically, the value of **Hu** should be set to make unnumbered headings (if any) be the lowest-level headings in a document.

The **HU** macro can be especially helpful in setting up appendices and other sections that may not fit well into the numbering scheme of the main body of a document {§O.2.1}.

D.4 Headings and Table of Contents

The text of headings and their corresponding page numbers can be automatically collected for a table of contents. This is accomplished by doing the following:

- setting the contents level register **Cl** to the highest heading level to be saved
- calling the **TC** macro {§K.1} at the end of the document.

Any heading whose level is less than or equal to the value of the **Cl** register is saved and later displayed in the table of contents. The default value for the **Cl** register is 2, i.e., the first two levels of headings are saved.

D.5 First-Level Headings and Page Numbering Style

By default, pages are numbered sequentially at the top of the page. For large documents, it may be desirable to use page numbering of the ''section-page'' form where ''section'' is the number of the current first-level heading. This page numbering style can be achieved by specifying the `-rN3` or `-rN5` flag on the command line {§J.3}. As a side effect, this also has the effect of setting *Ej* to 1, i.e., each first level section begins on a new page. In this style, the page number is printed at the bottom of the page so that the correct section number is printed.

D.6 User Exit Macros

This section is intended primarily for users who are accustomed to writing formatter macros.

`.HX` *dlevel rlevel heading-text*
`.HY` *dlevel rlevel heading-text*
`.HZ` *dlevel rlevel heading-text*

The **HX**, **HY**, and **HZ** macros are the means by which the user obtains a final level of control over the previously described heading mechanism. These macros are not defined by MM—they are intended to be defined by the user. The `.H` macro call calls **HX** shortly before the actual heading text is printed; it calls **HZ** as its last action. After **HX** is called, the size of the heading is calculated. This processing causes certain features that may have been included in **HX**, such as `.ti` for temporary indent, to be lost. After the size calculation, **HY** is called so that the user may respecify these features. All default actions occur if these macros are not defined. If **HX**, **HY**, or **HZ** are defined by the user, user-supplied definition is interpreted at the appropriate point. These macros can therefore influence handling of all headings because the**HU** macro is actually a special case of the **H** macro.

If the user originally called the **H** macro, then the derived level argument (*dlevel*) and the real level argument (*rlevel*) are both equal to the level given in the `.H` call. If the user originally called the **HU** macro {§D.3}, *dlevel* is equal to the contents of register **Hu**, and *rlevel* is 0. In both cases, *heading-text* is the text of the original invocation.

By the time **H** calls **HX**, it has already incremented the heading counter of the specified level {§D.2.2.5}, produced blank lines (vertical spaces) to precede the heading {§D.2.2.1}, and accumulated the ''heading mark'', i.e., the string of digits, letters, and periods needed for a numbered heading. When **HX** is called, all user-accessible registers and strings can be referenced, as well as the following:

string **}0** — If *rlevel* is nonzero, this string contains the ''heading mark''. Two unpaddable spaces (to separate the *mark* from the *heading*) have been appended to this string.
If *rlevel* is 0, this string is null.

register **;0** — This register indicates the type of spacing that is to follow the heading {§D.2.2.2}.
A value of 0 means that the heading is run-in.
A value of 1 means a break (but no blank line) is to follow the heading.
A value of 2 means that a blank line (one-half a vertical space) is to follow the heading.

string **}2** If ‘‘register ;0’’ is 0, this string contains two unpaddable spaces that will be used to separate the (run-in) heading from the following text.
If ‘‘register ;0’’ is nonzero, this string is null.

register **;3** This register contains an adjustment factor for a `.ne` request issued before the heading is actually printed. On entry to **HX**, it has the value 3 if *dlevel* equals 1, and 1 otherwise. The `.ne` request is for the following number of lines: the contents of the ‘‘register ;0’’ taken as blank lines (halves of vertical space) plus the contents of ‘‘register ;3’’ as blank lines (halves of vertical space) plus the number of lines of the heading.

The user may alter the values of }0, }2, and ;3 within **HX**. The following are examples of actions that might be performed by defining **HX** to include the lines shown:

- Change first-level heading mark from format *n*. to *n*.0:
  `.if \\$1=1 .ds }0 \\n(H1.0\`<sp>`\`<sp>
  (where <sp> stands for a space)
- Separate run-in heading from the text with a period and two unpaddable spaces:
  `.if \\n(;0=0 .ds }2 .\`<sp>`\`<sp>
- Assure that at least 15 lines are left on the page before printing a first-level heading:
  `.if \\$1=1 .nr ;3 15v-\\n(;0v`
- Add three additional blank lines before each first-level heading:
  `.if \\$1=1 .sp 3`
- Indent level 3 run-in headings by five spaces:
  `.if \\$1=3 .ti 5n`

If temporary strings or macros are used within **HX**, their names should be chosen with care {§O.1}.

When the **HY** macro is called after the `.ne` is issued, certain features requested in **HX** must be repeated.

For example:

```
.de HY
.if \\$1=3 .ti 5n
..
```

The **HZ** macro is called at the end of **H** to permit user-controlled actions after the heading is produced. In a large document, sections may correspond to chapters of a book; and the user may want to change a page header or footer, e.g.:

```
.de HZ
.if \\$1=1 .PF ‘‘Section \\$3’’
..
```

D.7 Hints for Large Documents

A large document is often organized for convenience into one input text file per section. If the files are numbered, it is wise to use enough digits in the names of these files for the maximum number of sections, i.e., use suffix numbers 01 through 20 rather than 1 through 9 and 10 through 20.

Users often want to format individual sections of long documents. To do this with the correct section numbers, it is necessary to set register **H1** to *one less* than the number of the section just before the corresponding `.H 1` call. For example, at the beginning of Part 5, insert

```
.nr H1 4
```

E. Lists

This part describes different styles of lists; automatically numbered and alphabetized lists, bullet lists, dash lists, lists with arbitrary marks, and lists starting with arbitrary strings, i.e., with terms or phrases to be defined.

E.1 List Macros

In order to avoid repetitive typing of arguments to describe the style or appearance of items in a list, MM provides a convenient way to specify lists. All lists share the same overall structure and are composed of the following basic parts:

- A *list-initialization macro* (`.AL .BL`, `.DL`, `.ML`, `.RL`, or `.VL`) determines the style of list: line spacing, indention, marking with special symbols, and numbering or alphabetizing of list items {§E.1.1}.
- One or more *list-item macros* (`.LI`) identifies each unique item to the system. It is followed by the actual text of the corresponding list item {§E.1.2}.
- The *list-end macro* (`.LE`) identifies the end of the list. It terminates the list and restores the previous indention {§E.1.3}.

Lists may be nested up to six levels. The list-initialization macro saves the previous list status (indention marking style, etc.); the `.LE` macro restores it.

With this approach, the format of a list is specified only once at the beginning of the list. In addition, by building onto the existing structure, users may create their own customized sets of list macros with relatively little effort ({§E.2} and {§E.3}).

E.1.1 List-Initialization Macros

List-initialization macros are implemented as calls to the more basic **LB** macro {§E.2}. They are:

`.AL`	Automatically Numbered or Alphabetized List {§E.1.1.1}
`.BL`	Bullet List {§E.1.1.2}
`.DL`	Dash List {§E.1.1.3}
`.ML`	Marked List {§E.1.1.4}
`.RL`	Reference List {§E.1.1.5}
`.VL`	Variable-Item List {§E.1.1.6}

E.1.1.1 Automatically Numbered or Alphabetized List

`.AL` [*type*] [*text-indent*] [`1`]

The **AL** macro is used to begin sequentially numbered or alphabetized lists. If there are no arguments, the list is numbered; and text is indented by *Li* (initially six) spaces from the indent in force when the .AL is called. This leaves room for a space, two digits, a period, and two spaces before the text. Values that specify indention must be unscaled and are treated as ‘‘character positions’’, i.e., number of ens.

Spacing at the beginning of the list and between items can be suppressed by setting the list space register (*Ls*). The*Ls* re gister is set to the innermost list level for which spacing is done. For example:

```
.nr Ls 0
```

specifies that no spacing will occur around any list items. The default value for *Ls* is six (which is the maximum list nesting level).

- The *type* argument may be given to obtain a different type of sequencing. Its value indicates the first element in the sequence desired. If *type* argument is omitted or null, the value 1 is assumed.

Argument	*Interpretation*
1	Arabic (default for all levels)
A	Uppercase alphabetic
a	Lowercase alphabetic
I	Uppercase Roman
i	Lowercase Roman

- If *text-indent* argument is non-null, it is used as the number of spaces from the current indent to the text; i.e., it is used instead of the *Li* register for this list only. If*te xt-indent* argument is null, the value of *Li* will be used.
- If the third argument is given, a blank line (one-half a vertical space) will not separate items in the list. A blank line will occur before the first item however.

E.1.1.2 Bullet List

`.BL` [*text-indent*] [`1`]

The **BL** macro begins a bullet list. Each list item is marked by a bullet (•) followed by one space.

- If the *text-indent* argument is non-null, it overrides the default indention (the amount of paragraph indention as given in the **Pi** register {§D.1.1}). In the default case, the text of a bullet list lines up with the first line of indented paragraphs.
- If the second argument is specified, no blank lines will separate items in the list.

E.1.1.3 Dash List

`.DL` [*text-indent*] [`1`]

The **DL** macro begins a dash list. Each list item is marked by a dash (“—”) followed by one space.

- If the *text-indent* argument is non-null, it overrides the default indention (the amount of paragraph indention as given in the **Pi** register {§D.1.1}). In the default case, the text of a dash list lines up with the first line of indented paragraphs.
- If the second argument is specified, no blank lines will separate items in the list.

E.1.1.4 Marked List

`.ML` *mark* [*text-indent*] [`1`]

The **ML** macro is much like **BL** and **DL** macros but expects the user to specify an arbitrary *mark*, which may consist of more than a single character.

- Text is indented *text-indent* spaces if the second argument is not null; otherwise, the text is indented one more space than the width of *mark*.
- If the third argument is specified, no blank lines will separate items in the list.

The *mark* must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified {§C.3}.

E.1.1.5 Reference List

`.RL` [*text-indent*] [`1`]

A **RL** macro call begins an automatically numbered list in which the numbers are enclosed by square brackets (**[ ]**).

- If *text-indent* argument is non-null, it is used as the number of spaces from the current indent to the text; i.e., it is used instead of **Li** for this list only. If*te xt-indent* argument is omitted or null, the value of **Li** is used.
- If the second argument is specified, no blank lines will separate the items in the list.

E.1.1.6 Variable-Item List

`.VL` *text-indent* [*mark-indent*] [`1`]

When a list begins with a **VL** macro, there is effectively no current *mark*; it is expected that each `.LI` will provide its own mark. This form is typically used to display definitions of terms or phrases.

- *Text-indent* provides the distance from current indent to beginning of the text.
- *Mark indent* produces the number of spaces from current indent to beginning of the *mark*, and it defaults to 0 if omitted or null.
- If the third argument is specified, no blank lines will separate items in the list.

An example of **VL** macro usage is shown below:

```
.tr ~
.VL 20 2
.LI mark~1
Here is a description of mark 1;
‘‘mark 1’’ of the .LI line contains a tilde
translated to an unpaddable space in order
to avoid extra spaces between
‘‘mark’’ and ‘‘1’’ {\(scC.3}.
.LI second~mark
This is the second mark also using a tilde translated
to an unpaddable space.
.LI third~mark~longer~than~indent:
This item shows the effect of a long mark; one space
separates the mark from the text.
.LI ~
This item effectively has no mark because the
tilde following the
.LI
is translated into a space.
.LE
```

when formatted yields:

mark 1
: Here is a description of mark 1; ‘‘mark 1’’ of the .LI line contains a tilde translated to an unpaddable space in order to avoid extra spaces between ‘‘mark’’ and ‘‘1’’ {§C.3}.

second mark
: This is the second mark also using a tilde translated to an unpaddable space.

third mark longer than indent:
: This item shows the effect of a long mark; one space separates the mark from the text.

  This item effectively has no mark because the tilde following the .LI is translated into a space.

The tilde argument on the last `.LI` above is required; otherwise, a ‘‘hanging indent’’ would have been produced. A ‘‘hanging indent’’ is produced by using `.VL` and calling **LI** with no arguments or with a null first argument. For example:

```
.VL 10
.LI
Here is some text to show a hanging indent.
The first line of text is at the left margin.
The second is indented 10 spaces.
.LE
```

when formatted yields:

Here is some text to show a hanging indent.
: The first line of text is at the left margin. The second is indented 10 spaces.

The *mark* must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified {§C.3}.

E.1.2 List-Item Macro

`.LI` [*mark*] [`1`]
one or more lines of text that make up the list item.

The **LI** macro is used with all lists and for each list item. It normally causes output of a single blank line (one-half a vertical space) before its list item although this may be suppressed.

- If no arguments are given, `.LI` labels the item with the current *mark* which is specified by the most recent list-initialization macro.
- If a single argument is given, that argument is output instead of the current *mark*.
- If two arguments are given, the first argument becomes a prefix to the current *mark* thus allowing the user to emphasize one or more items in a list. One unpaddable space is inserted between the prefix and the mark.

For example:

```
.BL 6
.LI
This is a simple bullet item.
.LI +
This replaces the bullet with a ‘‘plus’’.
.LI + 1
This uses a ‘‘plus’’ as prefix to the bullet.
.LE
```

when formatted yields:

- This is a simple bullet item.

+ This replaces the bullet with a ‘‘plus’’.

+ • This uses a ‘‘plus’’ as prefix to the bullet.

The *mark* must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified {§C.3}.

If the current *mark* (in the current list) is a null string and the first argument of .LI is omitted or null, the resulting effect is that of a ‘‘hanging indent’’, i.e., the first line of the following text is moved to the left starting at the same place where *mark* would have started {§E.1.1.6}.

E.1.3 List-End Macro

`.LE [ 1 ]`

The **LE** macro restores the state of the list to that existing just before the most recent list-initialization macro call. If the optional argument is given, the `.LE` outputs a blank line (one-half a vertical space). This option should generally be used only when the `.LE` is followed by running text but not when followed by a macro that produces blank lines of its own such as the **P**, **H**, or **LI** macro.

The **H** and **HU** macros automatically clear all list information. The user may omit the `.LE` macro calls that would normally occur just before either of the heading macros and not receive the ‘‘LE:mismatched’’ error message. Such a practice is not recommended because errors will

occur if the list text is separated from the heading at some later time (e.g., by insertion of text).

E.1.4 Example of Nested Lists

An example of input for the several lists and the corresponding output is shown below. The .AL and .DL macro calls {§E.1.1} contained therein are examples of list-initialization macros.

Input text is:

```
.AL A
.LI
This is alphabetized list item A.
This text shows the alignment of the second line
of the item.
Notice the text indentions and alignment of left
and right margins.
.AL
.LI
This is numbered item 1.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog’s back.
.DL
.LI
This is a dash item.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog’s back.
.LI + 1
This is a dash item with a ‘‘plus’’ as prefix.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog’s back.
.LE
.LI
This is numbered item 2.
.LE
.LI
This is another alphabetized list item B.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog’s back.
.LE
.P
This paragraph follows a list item and is aligned with
the left margin.
A paragraph following a list resumes the normal line
length and margins.
```

The output is:

A. This is alphabetized list item A. This text shows the alignment of the second line of the item. Notice the text indentions and alignment of left and right margins.

1. This is numbered item 1. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

   — This is a dash item. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

   + — This is a dash item with a ''plus'' as prefix. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

2. This is numbered item 2.

B. This is another alphabetized list item B. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

This paragraph follows a list item and is aligned with the left margin. A paragraph following a list resumes the normal line length and margins.

E.2 List-Begin Macro and Customized Lists

.LB *text-indent mark-indent pad type* [*mark*] [*LI-space*] [*LB-space*]

List-initialization macros described above suffice for almost all cases. However, if necessary, the user may obtain more control over the layout of lists by using the basic list-begin macro (**LB**). The**LB** macro is used by the other list-initialization macros. Its ar guments are as follows:

- The *text-indent* argument provides the number of spaces that text is to be indented from the current indent. Normally, this value is taken from the **Li** register (for automatic lists) or from the **Pi** register (for bullet and dash lists).
- The combination of *mark-indent* and *pad* arguments determines the placement of the mark. The mark is placed within an area (called *mark area*) that starts *mark-indent* spaces to the right of the current indent and ends where the text begins (i.e., ends *text-indent* spaces to the right of the current indent). The *mark-indent* argument is typically 0.
- Within the *mark area*, the mark is left justified if the *pad* argument is 0. If *pad* is a number *n* (greater than 0) then *n* blanks are appended to the mark; the *mark-indent* value is ignored. The resulting string immediately precedes the text. The*mark* is ef fectively right justified *pad* spaces immediately to the left of text.
- Arguments *type* and *mark* interact to control the type of marking used. If *type* is 0, simple marking is performed using the mark character(s) found in the *mark* argument. If*type* is greater than 0, automatic numbering or alphabetizing is done; and *mark* is then interpreted as the first item in the sequence to be used for numbering or alphabetizing, i.e., it is chosen from the set (1, A, a, I, i) as in {§E.1.1.1}. This is summarized below:

Argument		*Result*
type	*mark*	
0	omitted	hanging indent
0	*string*	*string* is the mark
>0	omitted	Arabic numbering
>0	one of: 1, A, a, I, i	automatic numbering or alphabetic sequencing

Each nonzero value of *type* from one to six selects a different way of displaying the marks. The following table shows the output appearance for each value of *type* :

Value	*Appearance*
1	*x.*
2	*x*)
3	(*x*)
4	[*x*]
5	< *x* >
6	{ *x* }

where *x* is the generated number or letter.

The *mark* must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified {§C.3}.

- The *LI-space* argument gives the number of blank lines (halves of a vertical space) that should be output by each .LI macro in the list. If omitted, *LI-space* defaults to 1; the value 0 can be used to obtain compact lists. If *LI-space* is greater than 0, the .LI macro issues a **.ne** request for two lines just before printing the mark.
- The *LB-space* argument is the number of blank lines (one-half a vertical space) to be output by .LB itself. If omitted *LB-space* defaults to 0.

There are three combinations of *LI-space* and *LB-space*:

- The normal case is to set *LI-space* to 1 and *LB-space* to 0 yielding one blank line before each item in the list; such a list is usually terminated with a .LE 1 macro to end the list with a blank line.
- For a more compact list, *LI-space* is set to 0, *LB-space* is set to 1, and the .LE 1 macro is used at the end of the list. The result is a list with one blank line before and after it.
- If both *LI-space* and *LB-space* are set to 0 and the .LE macro is used to end the list, a list without any blank lines will result.

Section E.3 shows how to build upon the supplied list of macros to obtain other kinds of lists.

E.3 User-Defined List Structures

This part is intended for users accustomed to writing formatter macros.

If a large document requires complex list structures, it is useful to define the appearance for each list level only once instead of having to define the appearance at the beginning of each list. This permits consistency of style in a large document. A generalized list-initialization macro might be defined in such a way that what the macro does depends on the list-nesting level in effect at the time the macro is called. Levels 1 through 5 of the lists to be formatted may have the following appearance:

A.

[1]

•

a)

+

The following code defines a macro (**aL**) that always begins a new list and determines the type of list according to the current list level. To understand it, the user should know that the number

register *:g* is used by the MM list macros to determine the current list level; it is 0 if there is no currently active list. Each call to a list-initialization macro increments *:g*, and each `.LE` call decrements it.

```
.\''register g is used as a local temporary to save :g
.de aL
.nr g \\n(:g
.if \\ng=0 .AL A                \''produces an A.
.if \\ng=1 .LB \\n(Li 0 1 4     \''produces a [1]
.if \\ng=2 .BL                  \''produces a bullet
.if \\ng=3 .LB \\n(Li 0 2 2 a   \''produces an a)
.if \\ng=4 .ML +                \''produces a +
..
```

This macro can be used (in conjunction with `.LI` and `.LE`) instead of `.AL`, `.RL`, `.BL`, `.LB`, and `.ML`. For example, the following input:

```
.aL
.LI
First line.
.aL
.LI
Second line.
.LE
.LI
Third line.
.LE
```

when formatted yields

A. First line.

   [1] Second line.

B. Third line.

There is another approach to lists that is similar to the **H** mechanism. List-initialization, as well as the .LI and the .LE macros, are all included in a single macro. That macro (defined as **bL** below) requires an argument to tell it what level of item is required; it adjusts the list level by

either beginning a new list or setting the list level back to a previous value, and then issues a .LI macro call to produce the item:

```
.de bL
.ie \\n(.$ .nr g \\$1           \''argument given, that is the level
.el .nr g \\n(:g                \''no argument, use current level
.if \\ng-\\n(:g>1 .)D ''ILLEGAL SKIPPING OF LEVEL''
.\''                              increasing level by more than 1
.if \\ng>\\n(:g \{.aL \\ng-1     \''if g > :g, begin new list
.nr g \\n(:g\}                  \''and reset g to current level
.\''                              (.aL changes g)
.if \\n(:g>\\ng .LC \\ng
.\''                              if :g > g, prune back to correct level
.\''                              if :g = g, stay within current list
.LI                             \''in all cases, get out an item
..
```

For **bL** to work, the previous definition of the **aL** macro must be changed to obtain the value of **g** from its argument rather than from **:g**. Invoking **bL** without arguments causes it to stay at the current list level.

The **LC** (List Clear) macro removes list descriptions until the level is less than or equal to that of its argument. For example, the **H** macro includes the call "`.LC 0`". If text is to be resumed at the end of a list, insert the call "`.LC 0`" to clear out the lists completely. The example below illustrates the relatively small amount of input needed by this approach. The input text

```
The quick brown fox jumped over the lazy dog's back.
.bL 1
First line.
.bL 2
Second line.
.bL 1
Third line.
.bL
Fourth line.
.LC 0
Fifth line.
```

when formatted yields:

The quick brown fox jumped over the lazy dog's back.

A. First line.

   [1] Second line.

B. Third line.

C. Fourth line.

Fifth line.

F. Memorandum and Released-Paper Documents

One use of MM is for the preparation of memoranda and released-paper documents (a documentation style used by Bell Laboratories) which have special requirements for the first page

and for the cover sheet. (An alternate style for business letters is discussed in the next section.) Data needed (title, author, date, case numbers, etc.) is entered the same for both styles; an argument to the .MT macro indicates which style is being used.

F.1 Sequence of Beginning Macros

Macros, if present, must be given in the following order:

```
.ND new-date
.TL [charging-case] [filing-case]
```
one or more lines of text
```
.AF [company-name]
.AU name [initials] [loc] [dept] [ext] [room] [arg] [arg]
.AT [title] . . .
.TM [number] . . .
.AS [arg] [indent]
```
one or more lines of abstract text
```
.AE
.NS [arg] [1]
```
one or more lines of ‘‘copy to’’ notation
```
.NE
.OK [keyword] . . .
.MT [type] [addressee]
```

The only required macros for a memorandum for file or a released-paper document are **TL**, **AU**, and **MT**; all other macros (and their associated input lines) may be omitted if the features are not needed. Once **MT** has been called, none of the above macros (except **NS** and **NE**) can be reinvoked because they are removed from the table of defined macros to save memory space.

If neither the memorandum nor released-paper style is desired, the **TL**, **AU**, **TM**, **AE**, **OK**, **MT**, **ND**, and **AF** macros should be omitted from the input text. If these macros are omitted, the first page will have only the page header followed by the body of the document.

The macros for memorandum and released-paper documents require the postprocessor **col** when you are using the *nroff* text formatter.

F.2 Title

```
.TL [charging-case] [filing-case]
```
one or more lines of title text

Arguments to the **TL** macro are the charging-case number(s) and filing-case number(s).

- The *charging-case* argument is the case number to which time was charged for the development of the project described in the memorandum. Multiple charging-case numbers are entered as ‘‘subarguments’’ by separating each from the previous with a comma and a space and enclosing the entire argument within double quotes.
- The *filing-case* argument is a number under which the memorandum is to be filed. Multiple filing case numbers are entered similarly. For example:

  ```
  .TL ‘‘12345, 67890’’ 987654321
  ```
  Construction of a Table of All Even Prime Numbers

The title of the memorandum or released-paper document follows the .TL macro and is processed in fill mode. The .br request may be used to break the title into several lines as follows:

```
.TL 12345
First Title Line
.br
\!.br
Second Title Line
```

On output, the title appears after the word ‘‘subject’’ in the memorandum style and is centered and printed in bold in the released-paper document style.

If only a charging case number or only a filing case number is given, it will be separated from the title in the memorandum style by a dash and will appear on the same line as the title. If both case numbers are given and are the same, then ‘‘Charging and Filing Case’’ followed by the number will appear on a line following the title. If the two case numbers are different, separate lines for ‘‘Charging Case’’ and ‘‘File Case’’ will appear after the title.

F.3 Authors

```
.AU name [initials] [loc] [dept] [ext] [room] [arg] [arg]
.AT [title] . . .
```

The **.AU** macro receives as arguments information that describes an author. If any argument contains blanks, that argument must be enclosed within double quotes. The first six arguments must appear in the order given. A separate .AU macro is required for each author.

The **.AT** macro is used to specify the author’s title. Up to nine arguments may be given. Each will appear in the signature block for memorandum style {§F.11} on a separate line following the signer’s name. The .AT must immediately follow the .AU for the given author. For example:

```
.AU ‘‘J. J. Jones’’ JJJ PY 9876 5432 1Z-234
.AT Director ‘‘Materials Research Laboratory’’
```

In the ‘‘from’’ portion in the memorandum style, the author’s name is followed by location and department number on one line and by room number and extension number on the next line. The ‘‘x’’ for the extension is added automatically. Printing of the location, department number, extension number, and room number may be suppressed on the first page of a memorandum by setting the register *Au* to 0; the default value for *Au* is 1. Arguments 7 through 9 of the .AU macro, if present, will follow this normal author information in the ‘‘from’’ portion, each on a separate line. These last three arguments may be used for organizational numbering schemes, etc. For example:

```
.AU ‘‘S. P. LeName’’ SPL IH 9988 7766 5H-444 543210MF
```

The name, initials, location, and department are also used in the signature block. Author information in the ‘‘from’’ portion, as well as names and initials in the signature block will appear in the same order as the .AU macros.

Names of authors in the released-paper style are centered below the title. Following the name of the last author, ‘‘Bell Laboratories’’ and the location are centered. The paragraph on memorandum types {§F.7} contains information regarding authors from different locations.

F.4 TM Numbers

```
.TM [number] . . .
```

If the memorandum is a technical memorandum, the TM numbers are supplied via the **.TM** macro. Up to nine numbers may be specified. For example:

```
.TM 7654321 77777777
```

This macro call is ignored in the released-paper and external-letter styles {§F.7}.

F.5 Abstract

```
.AS [arg] [indent]
```
text of abstract
```
.AE
```

If a memorandum has an abstract, the input is identified with the **.AS** (abstract start) and **.AE** (abstract end) delimiters. Abstracts are printed on page 1 of a document and/or on its cover sheet. There are three styles of cover sheet:

- Released paper
- Technical memorandum
- Memorandum for file {§K.2} (also used for engineer's notes, memoranda for record, etc.).

Cover sheets for released papers and technical memoranda are obtained by invoking the .CS macro {§K.2}.

In released-paper style (first argument of the .MT macro {§F.7} is 4) and in technical memorandum style if the first argument of .AS is:

0 − Abstract will be printed on page 1 and on the cover sheet (if any).

1 − Abstract will appear only on the cover sheet (if any).

In memoranda for file style and in all other documents (other than external letters) if the first argument of .AS is:

0 − Abstract will appear on page 1 and there will be no cover sheet printed.

2 − Abstract will appear only on the cover sheet which will be produced automatically (i.e., without invoking the .CS macro).

It is not possible to get either an abstract or a cover sheet with an external letter (first argument of the .MT macro is 5).

Notations such as a ''copy to'' list {§F.11.2} are allowed on memorandum for file cover sheets; the .NS and .NE macros must appear after the .AS 2 and .AE macros. Headings {§D.2, D.3} and displays {§H} are not permitted within an abstract.

The abstract is printed with ordinary text margins; an indention to be used for both margins can be specified as the second argument of .AS. Values that specify indention must be unscaled and are treated as ''character positions'', i.e., as the number of *ens*.

F.6 Other Keywords

```
.OK [ keyword ] ...
```

Topical keywords should be specified on a technical memorandum cover sheet. Up to nine such keywords or keyword phrases may be specified as arguments to the **.OK** macro; if any keyword contains spaces, it must be enclosed within double quotes.

F.7 Memorandum Types

```
.MT [type] [addressee]
```

The **MT** macro controls the format of the top part of the first page of a memorandum or of a released-paper document and the format of the cover sheets. The *type* arguments and corresponding values are:

type	*Value*
""	no memorandum type printed
0	no memorandum type printed
none	MEMORANDUM FOR FILE
1	MEMORANDUM FOR FILE
2	PROGRAMMER'S NOTES
3	ENGINEER'S NOTES
4	released-paper style
5	external-letter style
"*string*"	*string* (enclosed in quotes)

If the *type* argument indicates a memorandum style document, the corresponding statement indicated under ''VALUE'' will be printed after the last line of author information. If *type* is longer than one character, then the string, itself, will be printed. For example:

```
.MT ''Technical Note #5''
```

A simple letter is produced by calling .MT with a null (but not omitted) or 0 argument.

The second argument to .MT is the name of the addressee of a letter. If present, that name and the page number replace the normal page header on the second and following pages of a letter. For example:

```
.MT 1 ''John Jones''
```

produces

John Jones - 2

The *addressee* argument may not be used if the first argument is 4 (released-paper style document).

The released-paper style is obtained by specifying

```
.MT 4 [1]
```

This results in a centered, bold title followed by centered names of authors. The location of the last author is used as the location following ''Bell Laboratories'' (unless the .AF macro specifies a different company). If the optional second argument to .MT 4 is given, then the name of each author is followed by the respective company name and location. Information necessary for the memorandum style document but not for the released-paper style document is ignored.

If the released-paper style document is utilized, most Bell Laboratories (BTL) location codes are defined as strings that are the addresses of the corresponding BTL locations. These codes are needed only until the **MT** macro is called. Thus, following the .MT macro, the user may

reuse these string names. In addition, the macros for the end of a memorandum {§F.11} and their associated lines of input are ignored when the released-paper style is specified.

Authors from non-BTL locations may include their affiliations in the released-paper style by specifying the appropriate .AF macro {§F.9} and defining a string (with a 2-character name such as ZZ) for the address before each .AU. For example:

```
.TL
A Learned Treatise
.AF  ‘‘Getem Inc.’’
.ds  ZZ  22 Maple Avenue, Sometown 09999
.AU  ‘‘F. Swatter’’  ‘‘’’  ZZ
.AF  ‘‘Bell Laboratories’’
.AU  ‘‘Sam P. LeName’’  ‘‘’’  CB
.MT  4  1
```

In the external-letter style document (.MT 5), only the title (without the word ‘‘subject:’’) and the date are printed in the upper left and right corners, respectively, on the first page. It is expected that preprinted stationery will be used with the company logo and address of the author.

F.8 Date Changes/Removal

```
.ND new-date
```

The **.ND** macro alters the value of the string *DT*, which is initially set to produce the current date. If the argument contains spaces, it must be enclosed within double quotes. For example,

```
.ND ‘‘July 4, 1776’’
```

alters the date to be printed, and

```
.ND ‘‘’’
```

causes no date to be printed.

F.9 Alternate First-Page Format

```
.AF [company-name]
```

An alternate first-page format can be specified with the **.AF** macro. The words ‘‘subject’’, ‘‘date’’, and ‘‘from’’ (in the memorandum style) are omitted and an alternate company name is used.

If an argument is given, it replaces ‘‘Bell Laboratories’’ without affecting other headings. If the argument is null, ‘‘Bell Laboratories’’ is suppressed; and extra blank lines are inserted to allow room for stamping the document with a logo or a Bell Laboratories stamp.

The .AF with no argument suppresses ‘‘Bell Laboratories’’ and the ‘‘Subject/Date/From’’ headings, thus allowing output on preprinted stationery. The use of .AF with no arguments is equivalent to the use of -rA1 {§B.3}, except that the latter must be used if it is necessary to change the line length and/or page offset (which default to 5.8i and 1i, respectively, for preprinted forms). The command line options −rOk and −rWk {§B.3} are not effective with .AF. The only .AF use appropriate for the *troff* formatter is to specify a replacement for ‘‘Bell Laboratories’’.

The command line option −rE*n* {§B.3} controls the font of the ‘‘Subject/Date/From’’ block.

F.10 Example

Input text for a document may begin as follows:

```
.TL
MM - Memorandum Macros
.AU ''D. W. Smith'' DWS PY
.AU ''J. R. Mashey'' JRM PY
.AU ''E. C. Pariser (January 1992 Revision)'' ECP PY
.AU ''N. W. Smith (June 1992 Revision)'' NWS PY
.MT 4
```

Figures 5-3, 5-4, and 5-5 show the input text file and both the *nroff* and *troff* formatter outputs for a simple letter.

```
.ND ''May 31, 1992''
.TL 334455
Out-of-Hours Course Description
.AU ''D. W. Stevenson'' DWS PY 9876 5432 1X-123
.AF ''Your Company''
.MT 0
.DS
J. M. Jones:
.DE
.P
Please use the following description for the out-of-hours course
.I
Document Preparation on the UNIX*
.R
.FS *
Trademark of Bell Laboratories.
.FE
.I ''System:''
.P
The course is intended for clerks, typists, and others
who intend to use the UNIX system for preparing documentation.
The course will cover such topics as:
.VL 18
.LI Environment:
utilizing a time-sharing computer system;
accessing the system; using appropriate output terminals.
.LI Files:
how text is stored on the system; directories; manipulating files.
.LI Textediting:""
how to enter text so that subsequent revisions are easier to make;
how to use the editing system to add, delete, and move lines of text;
how to make corrections.
.LI Textprocessing:""
basic concepts; use of general purpose formatting packages.
.LI Otherfacilities:""
additional capabilities useful to the typist such as the \fIspell\fR,
\fIdiff\fR, and \fIgrep\fR commands,
and a desk-calculator package.
.LE
.SG jrm
.NS 0
S. P. LeName
I. M. Here
U. R. There
R. Rhoade
.NE
```

Figure 3. Example of Input For a Simple MM Letter

Your Company

subject: Out-of-Hours Course Description - Case 334455

date: May 31, 1992
from: D. W. Stevenson
PY 9876
1X-123 x5432

J. M. Jones:

Please use the following description for the out-of-hours course

Environment:	utilizing a time-sharing computer system; accessing the system; using appropriate output terminals.
Files:	how text is stored on the system; directories; manipulating files.
Text editing:	how to enter text so that subsequent revisions are easier to make; how to use the editing system to add, delete, and move lines of text; how to make corrections.
Text processing:	basic concepts; use of general-purpose formatting packages.
Other facilities:	additional capabilities useful to the typist such as the and commands, and a desk-calculator package.

PY-9876-DWS-jrm D. W. Stevenson

Copy to
S. P. LeName
I. M. Here
U. R. There
R. Rhoade

* Trademark of Bell Laboratories

Figure 4. Example of *Nroff* Output for a Simple Letter

Your Company

subject: **Out-of-Hours Course Description - Case 334455**

date: **May 31, 1992**
from: **D. W. Stevenson**
PY 9876
1X-123 x5432

J. M. Jones:

Please use the following description for the out-of-hours course *Document Preparation on the UNIX* System*:

Environment:	utilizing a time-sharing computer system; accessing the system; using appropriate output terminals.
Files:	how text is stored on the system; directories; manipulating files.
Text editing:	how to enter text so that subsequent revisions are easier to make; how to use the editing system to add, delete, and move lines of text; how to make corrections.
Text processing:	basic concepts; use of general-purpose formatting packages.
Other facilities:	additional capabilities useful to the typist such as the *spell*, *diff*, and *grep* commands, and a desk-calculator package.

PY-9876-DWS-jrm **D. W. Stevenson**

Copy to
S. P. LeName
I. M. Here
U. R. There
R. Rhoade

* Trademark of Bell Laboratories

Figure 5. Example of *Troff* Output for a Simple Letter

F.11 End of Memorandum Macros

At the end of a memorandum document (but not of a released-paper document), signatures of authors and a list of notations can be requested. The following macros and their input are ignored if the released-paper style document is selected.

F.11.1 Signature Block

```
.FC [closing]
.SG [arg] [1]
```

The **.FC** macro prints ‘‘Yours very truly,’’ as a formal closing, if no closing argument is used. It must be given before the .SG macro. A different closing may be specified as an argument to .FC.

The **.SG** macro prints the author’s name(s) after the formal closing, if any. Each name begins at the center of the page. Three blank lines are left above each name for the actual signature.

- If no arguments are given, the line of reference data (location code, department number, author's initials, and typist's initials, all separated by hyphens) will not appear.
- A non-null first argument is treated as the typist's initials and is appended to the reference data.
- A null first argument prints reference data without the typist's initials or the preceding hyphen.
- If there are several authors and if the second argument is given, reference data is placed on the line of the first author.

Reference data contains only the location and department number of the first author. Thus, if there are authors from different departments and/or from different locations, the reference data should be supplied manually after the invocation (without arguments) of the .SG macro.

For example:

```
.SG
.rs
.sp −1v
PY/MH-9876/5432-JJJ/SPL-cen
```

F.11.2 "Copy to" and Other Notations

```
.NS [arg] [1]
```
zero or more lines of the notation
```
.NE
```

Many types of notations (such as a list of attachments or ''Copy to'' lists) may follow signature and reference data. Various notations are obtained through the **.NS** macro, which provides for proper spacing and for breaking notations across pages, if necessary.

The optional second argument, if present, causes the first argument to be used as the *entire* notation string. Codes for *arg* and the corresponding notations are:

arg	*NOTATIONS*
none	Copy to
""	Copy to
0	Copy to
1	Copy (with att.) to
2	Copy (without att.) to
3	Att.
4	Atts.
5	Enc.
6	Encs.
7	Under Separate Cover
8	Letter to
9	Memorandum to
10	Copy (with atts.) to
11	Copy (without atts.) to
12	Abstract Only to
13	Complete Memorandum to
‘‘*string*’’	Copy (string) to
‘‘*string*’’, with 2nd arg	string

If *arg* consists of more than one character, it is placed within parentheses between the words ‘‘Copy’’ and ‘‘to’’.

For example:

```
.NS ‘‘with att. 1 only’’
```

will generate

Copy (with att. 1 only) to

as the notation.

More than one notation may be specified before the .NE macro because a .NS macro terminates the preceding notation, if any. For example:

```
.NS 4
Attachment 1-List of register names
Attachment 2-List of string and macro names
.NS 1
J. J. Jones
.NS 2
S. P. LeName
G. H. Hurtz
.NE
```

would be formatted as

Atts.
Attachment 1-List of register names
Attachment 2-List of string and macro names

Copy (with att.) to
J. J. Jones

Copy (without att.) to
S. P. LeName
G. H. Hurtz

If the second argument is used, then the first argument becomes the entire notation.

For example:

```
.NS ‘‘Table of Contents to’’ 1
```

would be formatted with

Table of Contents to

as the notation.

The .NS and .NE macros may also be used at the beginning following .AS 2 and .AE to place the notation list on the memorandum for file cover sheet {§F.5}. If notations are given at the beginning without .AS 2, they will be saved and output at the end of the document.

F.11.3 Approval Signature Line

```
.AV approver’s-name [1]
```

The **.AV** macro may be used after the last notation block to automatically generate a line with spaces for the approval signature and date. For example:

```
.AV ‘‘Jane Doe’’
```

produces

APPROVED:

_____

Jane Doe Date

The optional second argument, if present, prevents the ‘‘APPROVED:’’ mark from appearing above the approval line.

F.12 One-Page Letter

Letters which overflow onto two pages with only the signature, etc. on the second page may be squeezed to one page if desired. This is done by increasing the page length with the −rL*n* option, e.g., −rL90. This has the effect of making the formatter believe that the page is 90 lines long and therefore providing more space than usual to place the signature or the notations. This will work only for a single-page letter or memo.

G. Business Letters

An alternative to the formal memorandum/released-paper style is the business letter. Four types of business letters are available: blocked, semiblocked, full-blocked, and simplified.

G.1 Letter Type

```
.LT [arg]
```

The letter-type macros accepts one optional argument describing the overall style of the letter:

arg	*format*
none	**Blocked**. (Same as **BL**).
BL	**Blocked.** All lines except date, return address, closing, and writer's identification begin at left margin. These lines begin at the center of the line.
SB	**Semi-blocked.** Same as Blocked except each paragraph is indented 5 spaces.
FB	**Full-blocked.** All lines begin at the left margin.
SP	**Simplified.** Same as Full-blocked except that the salutation is replaced by an all-capital subject line and is followed by an additional blank line, the closing is omitted, the writer's identification is in all capital letters on one line.

The following table illustrates the placement of business letter components. (The components are discussed next):

Component	*Macro*	**BL**	**SB**	**FB**	**SP**
Confidential Notation	**.LO CN**	L	L	L	L
Reference Notation	**.LO RN**	C	C	L	L
Attention	**.LO AT**	L	L	L	L
Salutation	**.LO SA**	L	L	L	×
Subject Line	**.LO SJ**	L	I	L	L
Inside Address	**.IA/.IE**	L	L	L	L
Paragraph	**.P**	L	I	L	L
Formal Closing	**.FC**	C	C	L	L
Signature	**.SG**	C	C	L	L
Copy Notation	**.NS/.NE**	L	L	L	L
Writer's Address	**.WA/.WE**	C	C	L	L

Legend:
- C - Centered
- L - Left aligned
- I - Indented
- × - None

Note that use of **.LT** and **.MT** (memorandum style) are mutually exclusive— You may only use one or the other in a document. Any macros specific to the **.MT** style (.2C, .AF, .AS, .AT, .AU, .CS, .OK, .TC, .TL, .TM) are ignored if **.LT** is selected. Any macros specific to the **.LT** style (.WA, .WE, .IA, .IE, .LO) are ignored if **.MT** is selected.

The order of certain macros (WA and IA to be discussed below) is important. These macros must appear in this order in your document:

```
.ND (optional)
.WA
.WE
.IA
.IE
.LT
```

G.2 Writer's Address

```
.WA name [ title ]
Return Address
 ...
.WE
```

Use this macro to specify the writer (author) of the letter. For example:

```
.WA ‘‘James Lorrin, Ph.D.’’ Director
Summit Research Company
38 River Road
Summit, NJ 07901
.WE
```

The address may be omitted if no address is desired. Otherwise, it may not exceed 14 lines.

The two arguments to **.WA** provide information used by the **.SG** macro {§F.11.1}. If you do not use **.SG**, the writer's name will not appear on the letter.

If the letter has multiple writers (authors), each may be mentioned with the **.WA/.WE** macro, but only one (the first) may contain an address.

G.3 Inside Address

```
.IA
Address
...
.IE
```

or

```
.IA name
.IE
```

This macro is used to specify the addressee. If you wish to include only the addressee's name and no address, the second form above may be used. For example:

```
.IA
Christina Clark, Ph.D.
Elan Computer Group, Inc.
888 Villa Street
Mountain View, CA 94041
.IE
```

or

```
.IA ‘‘Christina Clark, Ph.D.’’
.IE
```

G.4 Letter Options

`.LO` *type* [*arg*]

This macro provides a method of adding five common business letter components, discussed next.

G.4.1 Confidential Notation

```
.LO CN
```

The word **CONFIDENTIAL** will be placed on the second line below the date in the left margin. The optional argument replaces the default word **CONFIDENTIAL**. For example,

```
.LO CN ‘‘RESTRICTED’’
```

prints **RESTRICTED** instead of **CONFIDENTIAL**.

G.4.2 Reference Notation

`.LO RN` *arg*

This will cause **In reference to:** followed by the *arg* to be printed two lines below the date line, left aligned with the date. A typical usage is:

```
.LO RN ‘‘Meeting of 6/23/92’’
```

G.4.3 Attention

`.LO AT` *name*

The word **ATTENTION:** and the argument *Name* are printed two lines below the inside address. For example:

```
.LO AT ‘‘Dr. Christina Clark, Director’’
```

G.4.4 Salutation

`.LO SA` “*Opening*”

The salutation option sets the opening greeting to *Opening*. For example:

```
.LO SA ‘‘Dear Dr. Clark,’’
```

If no *Opening* argument is given, the default **To Whom It May Concern:** is printed.

G.4.5 Subject Line

`.LO SJ` “*Subject*”

The word **SUBJECT:** followed by the argument *Subject* is printed two lines below the salutation line. An example is:

```
.LO SJ ‘‘Staff Meeting:’’
```

G.4.6 Closing a Letter

The signature block macros **.FC** and **.SG** {§F.11.1}, copy-to notations **.NS/.NE** {§F.11.2}, and approval line macro **.AV** {§F.11.3} may be used at the end of an **.LT** format business letter. These are described in the Memorandum Macros section {§F}.

G.5 Sample Business Letter

Here is a sample **LT** style letter. The input is:

```
.LO AT ‘‘Personnel’’
.WA ‘‘Bill Taylor’’ ‘‘Director of Placement Services’’
Columbia University
116th Street
New York, NY 10019
.WE
.LO SA ‘‘Dear Dr. Smith,’’
.LO RN ‘‘Career Day’’
.IA ‘‘Rebecca Smith’’
Business Computer Systems, Inc.
190 River Boulevard
Durham, NC 27707
.IE
.LT BL
.P
We would be happy to include a representative of your company
in our ‘‘Career Day’’ program this spring.
I am forwarding your request to James Blinn,
who is organizing the event this year.
.P
Thank you for your interest in our placement programs.
.FC ‘‘Sincerely,’’
.SG
.NS 5
.NE
.NS
J. Blinn
.NE
```

Figure 6. Sample Business Letter Input

The output is:

Columbia University
116th Street
New York, NY 10019
May 23, 1992

In reference to: Career Day

Rebecca Smith
Business Computer Systems, Inc.
190 River Boulevard
Durham, NC 27707

ATTENTION: Personnel

Dear Dr. Smith,

We would be happy to include a representative of your company in our "Career Day" program this spring. I am forwarding your request to James Blinn, who is organizing the event this year.

Thank you for your interest in our placement programs.

Sincerely,

Bill Taylor
Director of Placement Services

Enc.
Copy to
J. Blinn

Figure 7. Sample Business Letter Output

H. Displays

Displays are blocks of text that are to be kept together on a page and not split across pages. They are processed in an environment that is different from the body of the text (see the .ev request). The MM package provides two styles of displays: a *static* (.DS) style and a *floating* (.DF) style.

- In the *static* style, the display appears in the same relative position in the output text as it does in the input text. This may result in extra white space at the bottom of the page if the display is too long to fit in the remaining page space.
- In the *floating* style, the display ‘‘floats’’ through the input text to the top of the next page if there is not enough space on the current page. Thus input text that follows a floating display may precede it in the output text. A queue of floating displays is maintained so that their relative order of appearance in the text is not disturbed.

By default, a display is processed in no-fill mode with single spacing and is not indented from the existing margins. The user can specify indention or centering as well as fill-mode processing.

Displays and footnotes {§I} may never be nested in any combination. Although lists {§E} and paragraphs {§D.1} are permitted, no headings (.H or .HU) {§D.2, 4.3} can occur within displays or footnotes.

H.1 Static Displays

```
.DS [format] [fill] [rindent]
one or more lines of text
.DE
```

A static display is started by the **.DS** macro and terminated by the **.DE** macro. With no arguments, .DS accepts lines of text exactly as typed (no-fill mode) and will not indent lines from the prevailing left margin indention or from the right margin.

- The *format* argument is an integer or letter used to control the left margin indention and centering with the following meanings:

format	*MEANING*
""	no indent
0 or L	no indent
1 or I	indent by standard amount
2 or C	center each line
3 or CB	center as a block
omitted	no indent

- The *fill* argument is an integer or letter and can have the following meanings:

fill	*MEANING*
""	no-fill mode
0 or N	no-fill mode
1 or F	fill mode
omitted	no-fill mode

- The *rindent* argument is the number of characters that the line length should be decreased, i.e., an indention from the right margin. This number must be unscaled in the*nr off* formatter and is treated as *ens*. It may be scaled in the*tr off* formatter or else defaults to *ems*.

The standard amount of static display indention is taken from the *Si* register, a default value of five spaces. Thus, text of an indented display aligns with the first line of indented paragraphs, whose indent is contained in the *Pi* register {§D.1}. Even though their initial values are the same (default values), these two registers are independent.

The display *format* argument value 3 (or CB) centers (horizontally) the entire display as a block (as opposed to .DS 2 and .DF 2 which center each line individually). All collected lines are left justified, and the display is centered based on width of the longest line. This format must be used in order for the **eqn**/**neqn** ‘‘mark’’ and ‘‘lineup’’ feature to work with centered equations {§H.4}.

By default, a blank line (one-half a vertical space) is placed before and after *static* and *floating* displays. These blank lines before and after*static* displays can be inhibited by setting the re gister *Ds* to 0.

The following example shows usage of all three arguments for *static* displays. This block of text will be indented five spaces (ems in *troff*) from the left margin, filled, and indented five spaces (ems in *troff*) from the right margin (i.e., centered). The input text

```
.DS I F 5
‘‘We the people of the United States,
in order to form a more perfect union,
establish justice, ensure domestic tranquility,
provide for the common defense,
and secure the blessings of liberty to
ourselves and our posterity,
do ordain and establish this Constitution to the
United States of America.’’
.DE
```

produces

> ‘‘We the people of the United States, in order to form a more perfect union, establish justice, ensure domestic tranquility, provide for the common defense, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution to the United States of America.’’

H.2 Floating Displays

```
.DF [format] [fill] [rindent]
```
one or more lines of text
```
.DE
```

A floating display is started by the **.DF** macro and terminated by the **.DE** macro. Arguments have the same meanings as static displays described above, except indent, no indent, and centering are calculated with respect to the initial left margin. This is because prevailing indent may change between when the formatter first reads the floating display and when the display is printed. One blank line (one-half a vertical space) occurs before and after a floating display.

The user may exercise precise control over the output positioning of floating displays through the use of two number registers, *De* and *Df* (see below). When a floating display is encountered by the *nroff* or *troff* formatter, it is processed and placed onto a queue of displays waiting to be output. Displays are removed from the queue and printed in the order entered, which is the order they appeared in the input file. If a new floating display is encountered and the queue of displays is empty, the new display is a candidate for immediate output on the current page. Immediate output is governed by size of display and the setting of the *Df* register code. The *De* register code controls whether text will appear on the current page after a floating display has been produced.

As long as the display queue contains one or more displays, new displays will be automatically entered there, rather than being output. When a new page is started (or the top of the second column when in 2-column mode), the next display from the queue becomes a candidate for output if the *Df* register code has specified ‘‘top-of-page’’ output. When a display is output, it is also removed from the queue.

When the end of a section (using section-page numbering) or the end of a document is reached, all displays are automatically removed from the queue and output. This occurs before a .SG,

.CS, or .TC macro is processed.

A display will fit on the current page if there is enough room to contain the entire display or if the display is longer than one page in length and less than half of the current page has been used. A wide (full-page width) display will not fit in the second column of a 2-column document.

The *De* and *Df* number register code settings and actions are as follows:

De REGISTER

CODE	*ACTION*
0	No special action occurs (also the default condition).
1	A page eject will always follow the output of each floating display, so only one floating display will appear on a page and no text will follow it.

For any other code, the action performed is the same as for code 1.

Df REGISTER

CODE	*ACTION*
0	Floating displays will not be output until end of section (when section-page numbering) or end of document.
1	Output new floating display on current page if there is space; otherwise, hold it until end of section or document.
2	Output exactly one floating display from queue to the top of a new page or column (when in 2-column mode).
3	Output one floating display on current page if there is space; otherwise, output to the top of a new page or column.
4	Output as many displays as will fit (at least one) starting at the top of a new page or column. If the *De* register is set to 1, each display will be followed by a page eject causing a new top of page to be reached where at least one more display will be output.
5	Output a new floating display on the current page if there is room (default condition). Output as many displays (but at least one) as will fit on the page starting at the top of a new page or column. If the *De* register is set to 1, each display will be followed by a page eject causing a new top of page to be reached where at least one more display will be output.

For any code greater than 5, the action performed is the same as for code 5.

The .WC macro {§M.5} may also be used to control handling of displays in double-column mode and to control the break in text before floating displays.

H.3 Tables

```
.TS [H]
global options;
column descriptors.
title lines
[.TH [N]]
data within the table.
.TE
```

The **.TS** (table start) and **.TE** (table end) macros make possible the use of the **tbl** program. These macros are used to delimit text to be examined by **tbl** and to set proper spacing around the table. The display function and the **tbl** delimiting function are independent. In order to permit the user to keep together blocks that contain any mixture of tables, equations, filled text, unfilled text, and caption lines, the .TS/.TE block should be enclosed within a display (.DS/.DE). Floating tables may be enclosed inside floating displays (.DF/.DE).

Macros .TS and .TE permit processing of tables that extend over several pages. If a table heading is needed for each page of a multipage table, the ''H'' argument should be specified to the .TS macro as above. Following the options and format information, table title is typed on as many lines as required and is followed by the .TH macro. The .TH macro must occur when ''.TS H'' is used for a multipage table. This is not a feature of **tbl** but of the definitions provided by the MM macro package.

The **.TH** (table header) macro may take as an argument the letter **N**. This argument causes the table header to be printed only if it is the first table header on the page. This option is used when it is necessary to build long tables from smaller .TS H/.TE segments.

For example:

```
.TS H
global options;
column descriptors.
Title lines
.TH
data
.TE
.TS H
global options;
column descriptors.
Title lines
.TH N
data
.TE
```

will cause the table heading to appear at the top of the first table segment and no heading to appear at the top of the second segment when both appear on the same page. However, the heading will still appear at the top of each page that the table continues onto. This feature is used when a single table must be broken into segments because of table complexity (e.g., too many blocks of filled text). If each segment had its own .TS H/.TH sequence, it would have its own header. However, if each table segment after the first uses .TS H/.TH N, the table header will appear only at the beginning of the table and the top of each new page or column that the table continues onto.

For the *nroff* formatter, the −e option may be used for terminals, such as the 450, that are capable of finer printing resolution. This will cause better alignment of features such as the lines forming the corner of a box. The −e is not effective with **col**. (See the**TBL** chapter for more information on tables.)

H.4 Equations

```
.DS
.EQ [label]
```
equation(s)
```
.EN
.DE
```

Mathematical typesetting programs **eqn** and **neqn** expect to use the **.EQ** (equation start) and **.EN** (equation end) macros as delimiters in the same way that **tbl** uses .TS and .TE; however, .EQ and .EN must occur inside a .DS/.DE pair. There is an exception to this rule − if .EQ and .EN are used to specify only the delimiters for in-line equations or to specify **eqn**/**neqn** defines, the .DS and .DE macros must not be used; otherwise, extra blank lines will appear in the output.

The .EQ macro takes an argument that will be used as a label for the equation. By default, the label will appear at the right margin in the ‘‘vertical center’’ of the general equation. The *Eq* register may be set to 1 to change labeling to the left margin.

The equation will be centered for centered displays; otherwise, the equation will be adjusted to the opposite margin from the label.

H.5 Figure, Table, Equation, and Exhibit Titles

```
.FG [title] [override] [flag]
.TB [title] [override] [flag]
.EC [title] [override] [flag]
.EX [title] [override] [flag]
```

The **.FG** (figure title), **.TB** (table title), **.EC** (equation caption), and **.EX** (exhibit caption) macros are normally used inside .DS/.DE pairs to automatically number and title figures, tables, and equations. These macros use registers *Fg*, *Tb*, *Ec*, and *Ex*, respectively (see paragraph B.3 on −rN5 to reset counters in sections). For example:

```
.FG ‘‘This is a Figure Title’’
```

yields

Figure 1. This is a Figure Title

The .TB macro replaces ‘‘Figure’’ with ‘‘TABLE’’, the .EC macro replaces ‘‘Figure’’ with ‘‘Equation’’, and the .EX macro replaces ‘‘Figure’’ with ‘‘Exhibit’’. The output title is centered if it can fit on a single line; otherwise, all lines but the first are indented to line up with the first character of the title. The format of the numbers may be changed using the **.af** request of the formatter. By setting the*Of* re gister to 1, the format of the caption may be changed from

Figure 1. Title

to

Figure 1 − Title

The *override* argument is used to modify normal numbering. If the *flag* argument is omitted or 0, *override* is used as a prefix to the number; if the *flag* argument is 1, *override* is used as a suffix; and if the *flag* argument is 2, *override* replaces the number.

If −rN5 {§B.3} is given, ‘‘section-figure’’ numbering is set automatically and user-specified *override* argument is ignored. For example:

Figure 5-1. Title

As a matter of formatting style, table headings are usually placed above the text of tables, while figure, equation, and exhibit titles are usually placed below corresponding figures and equations.

H.6 List of Figures, Tables, Equations, and Exhibits

A list of figures, tables, exhibits, and equations are printed following the table of contents if the number registers *Lf, Lt, Lx,* and *Le* (respectively) are set to 1. The *Lf, Lt,* and *Lx* registers are 1 by default; *Le* is 0 by default {§S}.

Titles of these lists may be changed by redefining the following strings which are shown here with their default values:

```
.ds Lf LIST OF FIGURES
.ds Lt LIST OF TABLES
.ds Lx LIST OF EXHIBITS
.ds Le LIST OF EQUATIONS
```

I. Footnotes

There are two macros (.FS and .FE) that delimit text of footnotes, a string (F) that automatically numbers footnotes, and a macro (.FD) that specifies the style of footnote text. Footnotes are processed in an environment different from that of the body of text, refer to **.ev** request.

I.1 Automatic Numbering of Footnotes

Footnotes may be automatically numbered by typing the three characters ‘‘**F**’’ (i.e., invoking the string *F*) immediately after the text to be footnoted without any intervening spaces. This will place the next sequential footnote number (in a smaller point size) a half line above the text to be footnoted.

I.2 Delimiting Footnote Text

```
.FS [label]
```
one or more lines of footnote text
```
.FE
```

There are two macros that delimit the text of each footnote. The **.FS** (footnote start) macro marks the beginning of footnote text, and the **.FE** (footnote end) macro marks the end. The *label* on the .FS macro, if present, will be used to mark footnote text. Otherwise, the number retrieved from the string *F* will be used. Automatically numbered and user-labeled footnotes may be intermixed. If a footnote is labeled (.FS*label*), the te xt to be footnoted must be followed by *label*, rather than by ‘‘**F**’’. Text between .FS and .FE is processed in fill mode. Another .FS, a .DS, or a .DF are not permitted between .FS and .FE macros. If footnotes are required in the title, abstract, or table {§H.3}, only labeled footnotes will appear properly. Everywhere else automatically numbered footnotes work correctly.

For example:

Automatically numbered footnote:

```
This is the line containing the word\*F
.FS
This is the text of the footnote.
.FE
to be footnoted.
```

Labeled footnote:

```
This is a labeled*
.FS *
The footnote is labeled with an asterisk.
.FE
footnote.
```

Text of the footnote (enclosed within the .FS.FE pair) should immediately follow the word to be footnoted in the input text, so that ‘‘*F’’ or *label* occurs at the end of a line of input and the next line is the .FS macro call. It is also good practice to append an unpaddable space {§C.3} to ‘‘*F’’ or *label* when they follow an end-of-sentence punctuation mark (i.e., period, question mark, exclamation point).

The footnote label is defined via the **:p** number register, which may be reset, or style changed, if desired. Note that it should be set to the next footnote number minus the autoincrement value. For example,

```
.nr :p 98 2
```

would cause the footnotes following to be labeled 100, 102, 104, etc. Similarly, the style may be changed with **.af**. For example,

```
.af :p i
```

would cause footnotes to be printed in lower case roman numerals. (See **Number Register Requests** in the NROFF/TROFF Reference for details on **.af**.)

The next two sections illustrate the various available footnote styles as well as numbered and labeled footnotes.

I.3 Format Style of Footnote Text

```
.FD [arg] [1]
```

Within footnote text, the user can control formatting style by specifying text hyphenation, right margin justification, and text indention, as well as left or right justification of the label when text indenting is used. The **FD** macro is called to select the appropriate style.

The first argument (**arg**) is a number from the left column of the following table. Formatting style for each number is indicated in the remaining four columns. Further explanation of the first two of these columns is given in the definitions of the **.ad**, **.na**, **.hy**, and **.nh** (adjust, no adjust, hyphenation, and no hyphenation, respectively) requests.

arg	*Hyphenation*	*Adjust*	*Text Indent*	*Label Justification*
0	.nh	.ad	yes	left
1	.hy	.ad	yes	left
2	.nh	.na	yes	left
3	.hy	.na	yes	left
4	.nh	.ad	no	left
5	.hy	.ad	no	left
6	.nh	.na	no	left
7	.hy	.na	no	left
8	.nh	.ad	yes	right
9	.hy	.ad	yes	right
10	.nh	.na	yes	right
11	.hy	.na	yes	right

If the first argument to .FD is greater than 11, the effect is as if **.FD 0** were specified. If the first argument is omitted or null, the effect is equivalent to **.FD 10** in the *nroff* formatter and to **.FD 0** in the *troff* formatter; these are also the respective initial default values.

If the second argument is specified, then when a first-level heading is encountered, automatically numbered footnotes begin again with 1. This is most useful with the ‘‘section-page’’ page numbering scheme. As an example, the input line

```
.FD ‘‘‘‘ 1
```

maintains the default formatting style and causes footnotes to be numbered afresh after each first-level heading in a document.

Hyphenation across pages is inhibited by MM except for long footnotes that continue to the following page. If hyphenation is permitted, it is possible for the last word on the last line on the current page footnote to be hyphenated. The user has control over this situation by specifying an even .FD argument.

Footnotes are separated from the body of the text by a short line rule. Those that continue to the next page are separated from the body of the text by a full-width rule. In the *troff* formatter, footnotes are set in type two points smaller than the point size used in the body of text.

I.4 Spacing Between Footnote Entries

Normally, one blank line (a 3-point vertical space) separates footnotes when more than one occurs on a page. To change this spacing, the *Fs* number register is set to the desired value. For example:

```
.nr Fs 2
```

will cause two blank lines (a 6-point vertical space) to occur between footnotes.

```
.FD 10
.P
This example illustrates several footnote styles
for both labeled and automatically numbered footnotes.
With the footnote style set to the \fBnroff\fR default style,
the first footnote is processed\*F
.FS
This is the first footnote text example.
This is the default style (.FD 10) for the \fBnroff\fR formatter.
```

```
The right margin is not justified,
hyphenation is not permitted,
text is indented, and the automatically generated label is
right justified in the text-indent space.
.FE
and followed by a second footnote.*****
.FS *****
This is the second footnote text example.
This is also the \fBnroff\fR formatter default style (.FD 10)
but with a long footnote label (*****) provided by the user.
.FE
.FD 1
Footnote style is changed by using the .FD macro to
specify hyphenation, right margin justification,
indention, and left justification of the label.
This produces the third footnote,\*F
.FS
This is the third footnote example (.FD 1).
The right margin is justified, the footnote text is indented,
and the label is left justified in the text-indent space.
Although not necessarily illustrated by this example,
hyphenation is permitted.
.FE
and then the fourth footnote.\(dg
.FS \(dg
This is the fourth footnote example (.FD 1).
The style is the same as the third footnote.
.FE
.FD 6
Footnote style is set again via the .FD macro for no hyphenation,
no right margin justification,
no indention, and with the label left justified.
This produces the fifth footnote.\*F
.FS
This is the fifth footnote example (.FD 6).
The right margin is not justified, hyphenation is not permitted,
footnote text is not indented,
and the label is placed at the beginning of the first line.
.FE
```

Figure 8. Example of Input for Various Footnote Styles

This example illustrates several footnote styles for both labeled and automatically numbered footnotes. With the footnote style set to the *nroff* default style, the first footnote is processed[1] and followed by a second footnote.***** Footnote style is changed by using the .FD macro to specify hyphenation, right margin justification, indention, and left justification of the label. This

1. This is the first footnote text example. This is the default style (.FD 10) for the *nroff* formatter. The right margin is not justified, hyphenation is not permitted, text is indented, and the automatically generated label is right justified in the text-indent space.

***** This is the second footnote text example. This is also the*nr off* formatter default style (.FD 10) but with a long footnote label (*****) provided by the user.

produces the third footnote,[2] and then the fourth footnote.† Footnote style is set again via the .FD macro for no hyphenation, no right margin justification, no indention, and with the label left justified. This produces the fifth footnote.[3]

Figure 9. Example of Output for Various Footnote Styles

J. Page Headers and Footers

Text printed at the top of each page is called *page header*. Text printed at the bottom of each page is called *page footer*. There can be up to three lines of text associated with the header − every page, even page only, and odd page only. Thus the page header may have up to two lines of text − the line that occurs at the top of every page and the line for the even- or odd-numbered page. The same is true for the page footer.

This part describes the default appearance of page headers and page footers and ways of changing them. The term *header* (not qualified by *even* or *odd*) is used to mean the page header line that occurs on every page, and similarly for the term *footer*.

J.1 Default Headers and Footers

By default, each page has a centered page number as the header. There is no default footer and no even/odd default headers or footers except as specified in paragraph J.3.

In a memorandum or a released-paper style document, the page header on the first page is automatically suppressed provided a break does not occur before the .MT macro is called. Macros and text in the following categories do not cause a break and are permitted before the memorandum types (.MT) macro:

- Memorandum and released-paper style document macros (.TL, .AU, .AT, .TM, .AS, .AE, .OK, .ND, .AF, .NS, and .NE)
- Page headers and footers macros (.PH, .EH, .OH, .PF, .EF, and .OF)
- The **.nr** and **.ds** requests.

J.2 Header and Footer Macros

For header and footer macros (.PH .EH, .OH, .PF, .EF, and .OF) the argument [arg] is of the form:

" ’ *left-part* ’ *center-part* ’ *right-part* ’ "

If it is inconvenient to use apostrophe (**’**) as the delimiter because it occurs within one of the parts, it may be replaced uniformly by any other character. The**.fc** request redefines the delimiter. In formatted output, the parts are left justified, centered, and right justified, respectively. Any parts may be empty.

---

2. This is the third footnote example (.FD 1). The right margin is justified, the footnote text is indented, and the label is left justified in the text-indent space. Although not necessarily illustrated by this example, hyphenation is permitted.

† This is the fourth footnote example (.FD 1). The style is the same as the third footnote.

3. This is the fifth footnote example (.FD 6). The right margin is not justified, hyphenation is not permitted, footnote text is not indented, and the label is placed at the beginning of the first line.

J.2.1 Page Header

```
.PH [arg]
```

The **.PH** macro specifies the header that is to appear at the top of every page. The initial value is the default centered page number enclosed by hyphens. The page number contained in the*P* register is an Arabic number. The format of the number may be changed by the**.af** macro request.

If *"debug mode"* is set using the flag −rD1 on the command line {§B.3}, additional information printed at the top left of each page is included in the default header. This consists of the Source Code Control System (SCCS) release and level of MM (thus identifying the current version {§M.3}) followed by the current line number within the current input file.

J.2.2 Even-Page Header

```
.EH [arg]
```

The **.EH** macro supplies a line to be printed at the top of each even-numbered page immediately following the header. Initial value is a blank line.

J.2.3 Odd-Page Header

```
.OH [arg]
```

The **.OH** macro is the same as the .EH except that it applies to odd-numbered pages.

J.2.4 Page Footer

```
.PF [arg]
```

The **.PF** macro specifies the line that is to appear at the bottom of each page. Its initial value is a blank line. If the −rC*n* flag is specified on the command line {§B.3}, the type of copy follows the footer on a separate line. In particular, if −rC3 or −rC4 (DRAFT) is specified, the footer is initialized to contain the date {§F.8} instead of being a blank line.

J.2.5 Even-Page Footer

```
.EF [arg]
```

The **.EF** macro supplies a line to be printed at the bottom of each even-numbered page immediately preceding the footer. Initial value is a blank line.

J.2.6 Odd-Page Footer

```
.OF [arg]
```

The **.OF** macro supplies a line to be printed at the bottom of each odd-numbered page immediately preceding the footer. Initial value is a blank line.

J.2.7 First Page Footer

By default, the first page footer is a blank line. If, in the input text file, the user specifies .PF and/or .OF before the end of the first page of the document, these lines will appear at the bottom of the first page.

The header (whatever its contents) replaces the footer on the first page only if the −rN1 flag is specified on the command line {§B.3}.

J.3 Default Header and Footer With Section-Page Numbering

Pages can be numbered sequentially within sections by ‘‘section-number page-number’’ {§D.5}. To obtain this numbering style, −rN3 or −rN5 is specified on the command line. In this case, the default *footer* is a centered ‘‘section-page’’ number, e.g., 7-2; and the default page header is blank.

J.4 Strings and Registers in Header and Footer Macros

String and register names may be placed in arguments to header and footer macros. If the value of the string or register is to be computed when the respective header or footer is printed, invocation must be escaped by four backslashes. This is because string or register invocation will be processed three times:

1. as the argument to the header or footer macro
2. in a formatting request within the header or footer macro
3. in a **.tl** request during header or footer processing.

For example, if you wished to use the page number register *P* in the page header (.PH) macro, it would have to be escaped with four backslashes:

```
.PH ‘‘‘‘Page \\\\nP’’’
```

This creates a right-justified header containing the word ‘‘Page’’ followed by the page number.

For convenience, the percent (**%**) character is the same as the page number. Thus

```
.PH ‘‘‘‘Page %’’’
```

is equivalent to the above.

To specify a footer with the ‘‘section-page’’ style, the user specifies (see paragraph D.2.2.5 for meaning of *H1*), one might say:

```
.PF ‘‘‘‘– \\\\n(H1−\\\\nP –’’’
.PF ‘‘‘‘– \\\\n(H1−% –’’’
```

See also section **B.3** and the command line commands **-rN3** and **-rN5**.

If the user arranges for the string *a]* to contain the current section heading which is to be printed at the bottom of each page, the .PF macro call would be:

```
.PF ‘‘‘’\\\\*(a]’’’’
```

If only one or two backslashes were used, the footer would print a constant value for *a]*, namely, its value when .PF appeared in the input text.

J.5 Header and Footer Example

The following sequence specifies blank lines for header and footer lines, page numbers on the outside margin of each page (i.e., top left margin of even pages and top right margin of odd pages), and ‘‘Revision 3’’ on the top inside margin of each page (nothing is specified for the center):

```
.PH ''''
.PF ''''
.EH '''%''Revision 3'''
.OH '''Revision 3''%'''
```

J.6 Generalized Top-of-Page Processing

This part is intended only for users accustomed to writing formatter macros.

During header processing, MM calls two user-definable macros:

- The **.TP** (top of page) macro is called in the environment (refer to **.ev** request) of the header.
- The **.PX** is a page header user-exit macro that is called (without arguments) when the normal environment has been restored and with the ''no-space'' mode already in effect.

The effective initial definition of .TP (after the first page of a document) is

```
.de TP
.sp 3
.tl \\*(}t
.if e 'tl \\*(}e
.if o 'tl \\*(}o
.sp 2
..
```

The string *}t* contains the header, the string *}e* contains the even-page header, and the string *}o* contains the odd-page header as defined by the .PH, .EH, and .OH macros, respectively. To obtain more specialized page titles, the user may redefine the .TP macro to cause the desired header processing {§M.6}. Formatting done within the .TP macro is processed in an environment different from that of the body. For example, to obtain a page header that includes three centered lines of data, i.e., document number, issue date, and revision date, the user could define the .TP as follows:

```
.de TP
.sp
.ce 3
777-888-999
Iss. 2, AUG 1992
Rev. 7, SEP 1992
.sp
..
```

The .PX macro may be used to provide text that is to appear at the top of each page after the normal header and that may have tab stops to align it with columns of text in the body of the document.

J.7 Generalized Bottom-of-Page Processing

```
.BS
```
zero or more lines of text
```
.BE
```

Lines of text that are specified between the **.BS** (bottom-block start) and **.BE** (bottom-block end) macros will be printed at the bottom of each page after the footnotes (if any) but before the page footer. This block of text is removed by specifying an empty block, i.e.:

```
.BS
.BE
```

The bottom block will appear on the table of contents, pages, and the cover sheet for memorandum for file, but not on the technical memorandum or released-paper cover sheets.

J.8 Top and Bottom (Vertical) Margins

```
.VM [top] [bottom]
```

The **.VM** (vertical margin) macro allows the user to specify additional space at the top and bottom of the page. This space precedes the page header and follows the page footer. The .VM macro takes two unscaled arguments that are treated as v's. For example:

```
.VM 10 15
```

adds 10 blank lines to the default top of page margin and 15 blank lines to the default bottom of page margin. Both arguments must be positive (default spacing at the top of the page may be decreased by redefining .TP).

J.9 Proprietary Marking

```
.PM [code]
```

The **.PM** (proprietary marking) macro appends to the page footer a proprietary disclaimer. The *code* argument may be selected from the following table. Note that many *codes* may produce the same results (for compatibility with older versions of mm.) The codes in bold are the recommended codes to use.

CODE	*MESSAGE*
PM1,BP,N, P,BPN	*AT&T BELL LABORATORIES - PROPRIETARY* Use pursuant to G.E.I. 2.2
PM2,CA	THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF AT&T AND IS NOT TO BE DISCLOSED OR USED EXCEPT IN ACCORDANCE WITH APPLICABLE CONTRACTS OR AGREEMENTS.
PM3,CP	SEE PROPRIETARY NOTICE ON COVER PAGE
PM4,BPP, BR	*AT&T BELL LABORATORIES - PROPRIETARY (RESTRICTED)* Solely for authorized persons having a need to know pursuant to G.E.I. 2.2
PM5,ILL	THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF AT&T BELL LABORATORIES AND IS NOT TO BE DISCLOSED, REPRODUCED, OR PUBLISHED WITHOUT WRITTEN CONSENT. THIS DOCUMENT MUST BE RENDERED ILLEGIBLE WHEN BEING DISCARDED.
PM6,CI-II	*CI-II* Disclosure to AT&T Information Systems is subject to FCC separation requirements under Computer Inquiry II.

Figure 10. Proprietary Markings

Use the **.PM** macro at the beginning of your document before any footnotes are mentioned.

These disclaimers are in a form used by AT&T. You may change the text of these markings by editing the file **/usr/lib/macros/strings.mm**. If you wish to make temporary changes to a particular file only, simply include your own text for the *defined-strings* (**.ds**) as used in the **strings.mm** file. A copy of **strings.mm** with comments follows:

```
'''\"      The following string is used by the macro MT.
'''\"      ]S defined as logo character
.ds ]S \s36\(LH\s0
'''\"      }Z Company Name
.ds }Z AT&T Bell Laboratories
'''\"
'''\"      The following strings are used by the macro PM
'''\"
'''\"      Proprietary marking "PM1" (BP, N, P and BPN) lines 1-4
.ds ]M \fIAT&T BELL LABORATORIES - PROPRIETARY\fR
.ds ]O Use pursuant to G.E.I. 2.2
.ds ]Q
.ds ]R
'''\"      Proprietary marking "PM2" (CA) lines 1-4
.ds ]A THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF
.ds ]F AT&T AND IS NOT TO BE DISCLOSED OR USED EXCEPT IN
.ds ]G ACCORDANCE WITH APPLICABLE CONTRACTS OR AGREEMENTS.
.ds ]H
'''\"      Proprietary marking "PM3" (CP) lines 1-4
.ds ]I SEE PROPRIETARY NOTICE ON COVER PAGE
.ds ]J
.ds ]K
.ds ]L
'''\"      Proprietary marking "PM4" (BPP and BR) lines 1-4
.ds ]U \fIAT&T BELL LABORATORIES - PROPRIETARY (RESTRICTED)\fR
.ds ]V Solely for authorized persons having a need to know
.ds ]W pursuant to G.E.I. 2.2
.ds ]X
'''\"      Proprietary marking "PM5" (ILL) lines 1-4
.ds ]i THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF
.ds ]j AT&T BELL LABORATORIES AND IS NOT TO BE DISCLOSED,
.ds ]k REPRODUCED, OR PUBLISHED WITHOUT WRITTEN CONSENT.
.ds ]l THIS DOCUMENT MUST BE RENDERED ILLEGIBLE WHEN BEING DISCARDED.
'''\"      Proprietary marking "PM6" (CI-II) lines 1-4
.ds ]m \fICI-II\fR
.ds ]o Disclosure to AT&T Information Systems is subject
.ds ]p to FCC separation requirements under Computer Inquiry II.
.ds ]q
```

Figure 11. Proprietary Markings file ''strings.mm''

J.10 Private Documents

```
.nr Pv value
```

The word ''PRIVATE'' may be printed, centered, and underlined on the second line of a document (preceding the page header). This is done by setting the *Pv* register *value*:

value	*Meaning*
0	do not print PRIVATE (default)
1	PRIVATE on first page only
2	PRIVATE on all pages

If *value* is 2, the user definable **TP** macro may not be used because the **TP** macro is used by MM to print ''PRIVATE'' on all pages except the first page of a memorandum on which **TP** is not called.

K. Table of Contents and Cover Sheet

The table of contents and the cover sheet for a document are produced by invoking the .TC and .CS macros, respectively.

This section refers to cover sheets for technical memoranda and released papers only. The mechanism for producing a memorandum for file cover sheet was discussed earlier {§F.5}.

These macros normally appear once at the end of the document, after the Signature Block {§F.11.1} and Notations {§F.11.2} macros, and may occur in either order.

The table of contents is produced at the end of the document because the entire document must be processed before the table of contents can be generated. Similarly, the cover sheet may not be desired by a user and is therefore produced at the end.

K.1 Table of Contents

```
.TC [slevel] [spacing] [tlevel] [tab] [h1] [h2] [h3] [h4] [h5]
```

The **.TC** macro generates a table of contents containing heading levels that were saved for the table of contents as determined by the value of the *Cl* register {§D.4}. Arguments to .TC control spacing before each entry, placement of associated page number, and additional text on the first page of the table of contents before the word ‘‘CONTENTS’’.

Spacing before each entry is controlled by the first and second arguments (*slevel* and *spacing*). Headings whose level is less than or equal to *slevel* will have *spacing* blank lines (halves of a vertical space) before them. Both *slevel* and *spacing* default to 1. This means that first-level headings are preceded by one blank line (one-half a vertical space). The *slevel* argument does not control what levels of heading have been saved; saving of headings is the function of the *Cl* register.

The third and fourth arguments (*tlevel* and *tab*) control placement of associated page number for each heading. Page numbers can be justified at the right margin with either blanks or dots (called leaders) separating the heading text from the page number, or the page numbers can follow the heading text.

- For headings whose level is less than or equal to *tlevel* (default 2), page numbers are justified at the right margin. In this case, the value of *tab* determines the character used to separate heading text from page number. If*tab* is 0 (def ault value), dots (i.e., leaders) are used. If *tab* is greater than 0, spaces are used.
- For headings whose level is greater than *tlevel*, page numbers are separated from heading text by two spaces (i.e., page numbers are ‘‘ragged right’’, not right justified).

Additional arguments (*h1* . . . *h5*) are horizontally centered on the page and precede the table of contents.

If the **TC** macro is called with at most four arguments, the user-exit macro **TX** is called (without arguments) before the word ‘‘CONTENTS’’ is printed or the user-exit macro **TY** is called and the word ‘‘CONTENTS’’ is not printed.

By defining .TX or .TY and invoking .TC with at most four arguments, the user can specify what needs to be done at the top of the first page of the table of contents.

For example:

```
.de TX
.ce 2
Special Application
Message Transmission
.sp 2
.in +5n
Approved: \l'2.5i'
.in
.sp
..
.TC
```

yields the following output when the file is formatted

Special Application
Message Transmission

Approved: _____

CONTENTS

.
.
.

If the .TX macro were defined as .TY, the word ''CONTENTS'' would be suppressed. Defining .TY as an empty macro will suppress ''CONTENTS'' with no replacement:

```
.de TY
..
```

By default, the first level headings will appear in the table of contents left justified. Subsequent levels will be aligned with the text of headings at the preceding level. These indentions may be changed by defining the **Ci** string which takes a maximum of seven arguments corresponding to the heading levels. It must be given at least as many arguments as are set by the *Cl* register. Arguments must be scaled.

For example, with **Cl** = 5:

```
.ds Ci .25i .5i .75i 1i 1i  \troff"
```

or

```
.ds Ci 0 2n 4n 6n 8n  \nroff"
```

Two other registers are available to modify the format of the table of contents − *Oc* and *Cp*.

- By default, table of contents pages will have lowercase Roman numeral page numbering. If the *Oc* register is set to 1, the .TC macro will not print any page number but will instead reset the *P* register to 1. It is the user's responsibility to give an appropriate page footer to specify the placement of the page number. Ordinarily, the same .PF macro (page footer)

used in the body of the document will be adequate.

- The list of figures, tables, etc. pages will be produced separately unless *Cp* is set to 1 which causes these lists to appear on the same page as the table of contents.

K.2 Cover Sheet

```
.CS [pages] [other] [total] [figs] [tbls] [refs]
```

The **.CS** macro generates a cover sheet in either the released paper or technical memorandum style (see paragraph F.5 for details of the memorandum for file cover sheet). All other information for the cover sheet is obtained from data given before the .MT macro call {§F.1}. If the technical memorandum style is used, the .CS macro generates the ‘‘Cover Sheet for Technical Memorandum’’. The data that appear in the lower left corner of the technical memorandum cover sheet (counts of: pages of text, other pages, total pages, figures, tables, and references) are generated automatically (0 is used for ‘‘other pages’’). These values may be changed by supplying the corresponding arguments to the .CS macro. If the released-paper style is used, all arguments to .CS are ignored.

L. References

There are two macros (.RS and .RF) that delimit the text of references, a string that automatically numbers the subsequent references, and an optional macro (.RP) that produces reference pages within the document.

L.1 Automatic Numbering of References

Automatically numbered references may be obtained by typing ***(Rf** (invoking the string *Rf*) immediately after the text to be referenced. This places the next sequential reference number (in a smaller point size) enclosed in brackets one-half line above the text to be referenced. Reference count is kept in the *Rf* number register. The number register actually used to print the reference number for each reference call (*(Rf) in the text is *:R*. The*:R* re gister may have its format or value changed to effect the reference marks, without affecting the total count of references.

L.2 Delimiting Reference Text

```
.RS [string-name]
.RF
```

The **.RS** and **.RF** macros are used to delimit text of each reference as shown below:

```
A line of text to be referenced.\*(Rf
.RS
reference text
.RF
```

L.3 Subsequent References

The **.RS** macro takes one argument, a *string-name*. For example:

```
.RS aA
reference text
.RF
```

The string *aA* is assigned the current reference number. This string may be used later in the document as the string call, ***(***aA*, to reference text which must be labeled with a prior reference number. The reference is output enclosed in brackets one-half line above the text to be referenced. No .RS/.RF pair is needed for subsequent references.

L.4 Reference Page

```
.RP [arg1] [arg2]
```

A reference page, entitled by default ‘‘REFERENCES’’, will be generated automatically at the end of the document (before table of contents and cover sheet) and will be listed in the table of contents. This page contains the reference items (i.e., reference text enclosed within .RS/.RF pairs). Reference items will be separated by a space (one-half a vertical space) unless the *Ls* register is set to 0 to suppress this spacing. The user may change the reference page title by defining the *Rp* string:

```
.ds Rp New Title
```

The **.RP** (reference page) macro may be used to produce reference pages anywhere else within a document (i.e., after each major section). It is not needed to produce a separate reference page with default spacings at the end of the document.

Two .RP macro arguments allow the user to control resetting of reference numbering and page skipping.

arg1	*Meaning*
0	reset reference counter (default)
1	do not reset reference counter

arg2	*Meaning*
0	put on separate page (default)
1	do not cause a following .SK
2	do not cause a preceding .SK
3	no .SK before or after

If no .SK macro is issued by the .RP macro, a single blank line will separate the references from the following/preceding text. The user may wish to adjust spacing. For example, to produce references at the end of each major section:

```
.sp 3
.RP 1 2
.H 1 ‘‘Next Section’’
```

M. Miscellaneous Features

M.1 Bold, Italic, and Roman Fonts

```
.B [bold-arg] [previous-font-arg] . . .
.I [italic-arg] [previous-font-arg] . . .
.R
```

When called without arguments, the **.B** macro changes the font to bold and the **.I** macro changes to underlining (Italic). This condition continues until the occurrence of the **.R** macro which causes the Roman font to be restored. Thus:

```
.I
here is some text.
.R
```

yields underlined text via the *nroff* and Italic text via the *troff* formatter.

If the .B or .I macro is called with one argument, that argument is printed in the appropriate font (underlined in the *nroff* formatter for .I). Then the previous font is restored (underlining is turned off in the *nroff* formatter). If two or more arguments (maximum six) are given with a .B or .I macro call, the second argument is concatenated to the first with no intervening space (1/12 space if the first font is Italic) but is printed in the previous font. Remaining pairs of arguments are similarly alternated. For example:

```
.I italic ''<sp>text<sp>'' right -justified
```

where <sp> indicates a space, produces:

italic text *right*-justified

The .B and .I macros alternate with the prevailing font at the time the macros are called. To alternate specific pairs of fonts, the following macros are available:

.IB .BI .IR .RI .RB .BR

Each macro takes a maximum of six arguments and alternates arguments between specified fonts. For example,

```
.RB  ABCD  EFGH  IJKL  MNOP  QRST  UVWX
```

prints as

ABCD**EFGH**IJKL**MNOP**QRST**UVWX**

When using a terminal that cannot underline, the following can be inserted at the beginning of the document to eliminate all underlining:

```
.rm ul
.rm cu
```

Font changes in headings are handled separately {§D.2.2.4.1}.

M.2 Justification of Right Margin

```
.SA [arg]
```

The **.SA** macro is used to set right-margin justification for the main body of text. Two justification flags are used;*current* and *default*. Initially, both flags are set for no justification in the *nroff* formatter and for justification in the *troff* formatter. The argument causes the following action:

arg	*Meaning*
0	Sets both flags to no justification. It acts like the **.na** request.
1	Sets both flags to cause both right and left justification, the same as the **.ad** request.
omitted	Causes the current flag to be copied from the default flag, thus performing either a **.na** or **.ad** depending on the default condition.

`.SA` should be used to turn off and restore justification, rather than the `.ad` request.

M.3 SCCS Release Identification

The *RE* string contains the SCCS release and the MM text formatting macro package current version level. For example:

```
This is version \*(RE of the macros.
```

produces

This is version 10.129 of the macros.

This information is useful in analyzing suspected bugs in MM. The easiest way to have the release identification number appear in the output is to specify −rD1 {§B.3} on the command line. This causes the*RE* string to be output as part of the page header {§J.2.1}.

M.4 Two-Column Output

```
.2C
text and formatting requests (except another .2C)
.1C
```

The MM text formatting macro package can format two-columns on a page. The **.2C** macro begins 2-column processing which continues until a **.1C** macro (1-column processing) is encountered. In 2-column processing, each physical page is thought of as containing 2-columnar ‘‘pages’’ of equal (but smaller) ‘‘page’’ width. Page headers and footers are not affected by 2-column processing. The .2C macro does not balance 2-column output.

M.5 Footnotes and Displays for Two-Column Output

It is possible to have full-page width footnotes and displays when in 2-column mode, although default action is for footnotes and displays to be narrow in 2-column mode and wide in 1-column mode.

Footnote and display width is controlled by the **.WC** (width control) macro, which takes the following arguments:

arg	*MEANING*
N	Default mode (−WF, −FF, −WD, FB).
WF	Wide footnotes (even in 2-column mode).
−WF	DEFAULT: Turn off WF. Footnotes follow column mode; wide in 1-column mode (1C), narrow in 2-column mode (2C), unless FF is set.

FF First footnote. All footnotes have same width as first footnote encountered for that page.

−FF DEFAULT: Turn off FF. Footnote style follows settings of WF or −WF.

WD Wide displays (even in 2-column mode).

−WD DEFAULT: Displays follow the column mode in effect when display is encountered.

FB DEFAULT: Floating displays cause a break when output on the current page.

−FB Floating displays on current page do not cause a break.

The **.WC WD FF** command will cause all displays to be wide and all footnotes on a page to be the same width while **.WC N** will reinstate default actions. If conflicting settings are given to **.WC**, the last one is used. A **.WC WF −WF** command has the effect of a **.WC −WF**.

M.6 Column Headings for Two-Column Output

This section is intended only for users accustomed to writing formatter macros.

In 2-column processing output, it is sometimes necessary to have headers over each column, as well as headers over the entire page. This is accomplished by redefining the .TP macro {§J.6} to provide header lines both for the entire page and for each of the columns. For example:

```
.de TP
.sp 2
.tl 'Page %'OVERALL''
.tl ''TITLE''
.sp
.nf
.ta 16C 31R 34 50C 65R
leftcenterrightleftcenterright
first columnsecond column
.fi
.sp 2
..
```

where stands for the tab character.

The above example will produce two lines of page header text plus two lines of headers over each column. Tab stops are for a 65-en overall line length.

M.7 Vertical Spacing

```
.SP [lines]
```

There exists several ways of obtaining vertical spacing, all with different effects. The**.sp** request spaces the number of lines specified unless the no space (**.ns**) mode is on, then the **.sp** request is ignored. The no space mode is set at the end of a page header to eliminate spacing by a **.sp** or **.bp** request that happens to occur at the top of a page. This mode can be turned off by the **.rs** (restore spacing) request.

The **.SP** macro is used to avoid the accumulation of vertical space by successive macro calls. Several .SP calls in a row will not produce the sum of the arguments but only the maximum argument. For example, the following produces only three blank lines:

```
.SP 2
.SP 3
.SP
```

Many MM macros utilize .SP for spacing. For example, ‘‘.LE 1’’ {§E.1.3} immediately followed by ‘‘.P’’ {§D.1} produces only a single blank line (one-half a vertical space) between the end of the list and the following paragraph. An omitted argument defaults to one blank line (one vertical space). Negative arguments are not permitted. The argument must be unscaled but fractional amounts are permitted. The .SP macro (as well as **.sp**) is also inhibited by the **.ns** request.

M.8 Skipping Pages

```
.SK [pages]
```

The **.SK** macro skips pages but retains the usual header and footer processing. If the *pages* argument is omitted, null, or 0, .SK skips to the top of the next page unless it is currently at the top of a page (then it does nothing). A ‘‘.SK *n*’’ command skips *n* pages. A ‘‘.SK’’ positions text that follows it at the top of a page, while ‘‘.SK 1’’ leaves one page blank except for the header and footer.

M.9 Forcing an Odd Page

```
.OP
```

The **.OP** macro is used to ensure that formatted output text following the macro begins at the top of an odd-numbered page.

- If currently at the top of an odd-numbered page, text output begins on that page (no motion takes place).
- If currently on an even page, text resumes printing at the top of the next page.
- If currently on an odd page (but not at the top of the page), one blank page is produced, and printing resumes on the next odd-numbered page after that.

M.10 Setting Point Size and Vertical Spacing

```
.S [point size] [vertical spacing]
```

The prevailing point size and vertical spacing may be changed by invoking the **.S** macro. In the *troff* formatter, the default point size (obtained from the MM register *S* {§B.3}) is 10 points, and the vertical spacing is 12 points (six lines per inch). The mnemonics D (default value), C (current value), and P (previous value) may be used for both arguments.

- If an argument is *negative*, current value is decremented by the specified amount.
- If an argument is *positive*, current value is incremented by the specified amount.
- If an argument is *unsigned*, it is used as the new value.
- If there are no arguments, the .S macro defaults to P.
- If the first argument is specified but the second is not, then (default) D is used for the vertical spacing.

Default value for vertical spacing is always two points greater than the current point size. Footnotes {§I} are two points smaller than the body with an additional 3-point space between foot-

notes. A null ("") value for either argument defaults to C (current value). Thus, if *n* is a numeric value:

```
.S          =     .S P P
.S "" n     =     .S C n
.S n ""     =     .S n C
.S n        =     .S n D
.S ""       =     .S C D
.S "" ""    =     .S C C
.S n n      =     .S n n
```

If the first argument is greater than 99, the default point size (10 points) is restored. If the second argument is greater than 99, the default vertical spacing (current point size plus two points) is used. For example:

```
.S 100      =     .S 10 12
.S 14 111   =     .S 14 16
```

M.11 Reducing Point Size of a String

`.SM` *string1* [*string2*] [*string3*]

The **SM** macro allows the user to reduce by one point the size of a string. If the third argument (*string3*) is omitted, the first argument (*string1*) is made smaller and is concatenated with the second argument (*string2*) if specified. If all three arguments are present (even if any are null), the second argument is made smaller and all three arguments are concatenated. For example:

Input	*Output*
.SM X	X
.SM X Y	XY
.SM Y X Y	YXY
.SM YXYX	YXYX
.SM YXYX)	YXYX)
.SM (YXYX)	(YXYX)
.SM Y XYX ""	YXYX

M.12 Producing Accents

If the Roman-8 character set is available, producing characters with accents is simple—just use the already defined symbol for the particular character. Here is a list of Roman-8 characters and the *troff* input to print them:

Troff Input	*Symbol*	*Troff Input*	*Symbol*
\(..	¨	\(I´	Í
~	~	\(i`	ì
\(aa	´	\(I`	Ì
\´	´	\(iˆ	î
\(ga	`	\(Iˆ	Î
\`	`	\(L-	£
\(A:	Ä	\(L=	
\(a:	ä	\(n~	ñ
\(ae	æ	\(N~	Ñ
\(AE	Æ	\(o/	ø
\(ao	å	\(O/	Ø
\(Ao	Å	\(o:	ö
\(A´	Á	\(O:	Ö
\(a´	á	\(ox	¤
\(a`	à	\(o´	ó
\(A`	À	\(O´	Ó
\(aˆ	â	\(o`	ò
\(Aˆ	Â	\(O`	Ò
\(a_	ª	\(oˆ	ô
\(a~	ã	\(Oˆ	Ô
\(A~	Ã	\(o_	º
\(c,	ç	\(o~	õ
\(C,	Ç	\(O~	Õ
\(ct	¢	\(ss	ß
\(d-	ð	\(sv	š
\(D-	Ð	\(Sv	Š
\(e:	ë	\(TH	Þ
\(E:	Ë	\(th	þ
\(e´	é	\(u:	ü
\(E´	É	\(U:	Ü
\(e`	è	\(u´	ú
\(E`	È	\(U´	Ú
\(eˆ	ê	\(u`	ù
\(Eˆ	Ê	\(U`	Ù
\(i:	ï	\(uˆ	û
\(I:	Ï	\(Uˆ	Û
\(I!	¡	\(y:	ÿ
\(I?	¿	\(Y:	Ÿ
\(i´	í	\(Y=	¥

Figure 12. International (Roman-8) Characters

If the Roman-8 character set is not available, these symbols may be produced (with much more difficulty) by defined strings. Here are some examples:

	Input	*Output*
Grave accent	c*`	c̀
Acute accent	e*'	é
Tilde	n*~	ñ
Circumflex	o*^	ô
Cedilla	c*,	ç
Lower-case umlaut	u*:	ü
Upper-case umlaut	U*;	Ü

The string definitions that generated these letters are:

```
.ds ‘ \\k:\h@-\\n(.wu*8u/10u@\h@\\n(.fu/2u*2u+1u-\\n(.fu*.2m@\(ga\h@|\\n:u@
.ds ’ \\k:\h@-\\n(.wu*8u/10u@\h@\\n(.fu/2u*2u+1u-\\n(.fu*.2m+.07m@\(aa\h@|\\n:u@
.ds ^ \\k:\h@-\\n(.wu*87u/100u@\h@\\n(.fu/2u*2u+1u-\\n(.fu*.15m-.07m@\
\h@\\n(.fu-1u/2u*.02m@\v@.1m@^\v@-.1m@\h@|\\n:u@
.ds ~ \\k:\h@-\\n(.wu*8u/10u@\h@\\n(.fu/2u*2u+1u-\\n(.fu*.2m-.07m@\
\h@\\n(.fu-1u/2u*.05m@~\h@|\\n:u@
.ds , \\k:\h@-\\n(.wu*85u/100u@\v@.04m@,\v@-.04m@\h@|\\n:u@
.ds : \\k:\h@-\\n(.wu*85u/100u@\h@\\n(.fu/2u*2u+1u-\\n(.fu*3u*.06m@\
\h@3u-\\n(.fu/2u*.05m-.1m@\
\v@-.6m@\z.\h@\\n(.fu-1u/2u*.05m+.2m@.\v@.6m@\h@|\\n:u@
.ds ; \\k:\h@-\\n(.wu*75u/100u@\h@\\n(.fu/2u*2u+1u-\\n(.fu*3u*.09m@\
\h@3u-\\n(.fu/2u*.06m-.15m@\h@\\n(.fu-1u/2u*.04m@\
\v@-.85m@\z.\h@.3m@.\v@.85m@\h@|\\n:u@
```

M.13 Inserting Text Interactively

```
.RD [prompt] [diversion] [string]
```

The **.RD** (read insertion) macro allows a user to stop the standard output of a document and to read text from the standard input until two consecutive newline characters are found. When newline characters are encountered, normal output is resumed.

- The *prompt* argument will be printed at the terminal. If not given, .RD signals the user with a BEL on terminal output.
- The *diversion* argument allows the user to save all text typed in after the prompt in a macro whose name is that of the diversion.
- The *string* argument allows the user to save for later reference the first line following the prompt in the named string.

The **RD** macro follows the formatting conventions in effect. Thus, the following examples assume that **RD** is called in no fill mode (**.nf**):

```
.RD Name aA bB
```

produces

Name: J. Jones (user types name)
16 Elm Rd.,
Piscataway

The diverted macro .aA will contain

J. Jones
16 Elm Rd.,
Piscataway

The string *bB* (*(*bB*) contains ‘‘J. Jones’’.

A newline character followed by an EOF (user specifiable CONTROL d) also allows the user to resume normal output.

N. Errors and Debugging

N.1 Error Terminations

When a macro detects an error, the following actions occur:

- A break occurs.
- The formatter output buffer (which may contain some text) is printed to avoid confusion regarding location of the error.
- A short message is printed giving the name of the macro that detected the error, type of error, and approximate line number in the current input file of the last processed input line. Error messages are explained in the last section of this chapter.
- Processing terminates unless register D {§B.3} has a positive value. In the latter case, processing continues even though the output is guaranteed to be deranged from that point on.

The error message is printed by outputting the message directly to the user terminal. If an output filter, such as **300**, **450**, or **hp** is being used to post-process the *nroff* formatter output, the message may be garbled by being intermixed with text held in that filter’s output buffer.

N.2 Disappearance of Output

Disappearance of output usually occurs because of an unclosed diversion (e.g., a missing .DE or .FE macro). Fortunately, macros that use diversions are careful about it, and these macros check to make sure that illegal nestings do not occur. If any error message is issued concerning a missing .DE or .FE, the appropriate action is to search backwards from the termination point looking for the corresponding associated .DF, .DS, or.FS (since these macros are used in pairs).

The following command:

grep −n ’^\.[EDFRT][EFNQS]’ files . . .

prints all the .DF, .DS, .DE, .EQ, .EN, .FS, .FE, .RS, .RF, .TS, and .TE macros found in *files* . . ., each preceded by its file name and the line number in that file. This listing can be used to check for illegal nesting and/or omission of these macros.

O. Extending and Modifying MM Macros

O.1 Naming Conventions

In this part, the following conventions are used to describe names:

- n: digit
- a: lowercase letter
- A: uppercase letter
- x: any alphanumeric character (n, a, or A, i.e., letter or digit)
- s: any nonalphanumeric character (special character).

All other characters are literals (characters that stand for themselves).

Request, macro, and string names are kept by the formatters in a single internal table; therefore, there must be no duplication among such names. Number register names are kept in a separate table.

O.1.1 Names Used by Formatters

requests:	aa (most common) an (only one, currently: c2)
registers:	aa (normal) .x (normal) .s (only one, currently: .$) a. (only one, currently: c.) % (page number)

O.1.2 Names Used by MM

macros and strings:	A, AA, Aa (accessible to users; e.g., macros P and HU, strings F, BU, and Lt) nA (accessible to users; only two, currently: 1C and 2C) aA (accessible to users; only one, currently: nP) s (accessible to users; only the seven accents, currently {§M.10}))x, }x,]x, >x, ?x (internal)
registers:	An, Aa (accessible to users; e.g., H1, Fg) A (accessible to users; meant to be set on the command line; e.g., C) :x, ;x, #x, ?x, !x (internal)

O.1.3 Names Used by eqn/neqn, and tbl

Mathematical equation preprocessors, **eqn** and **neqn**, use registers and string names of the form *nn*. The table preprocessor, **tbl**, uses T&, T#, and TW, and names of the form:

a− a+ a| nn na ˆa #a #s

O.1.4 Names Defined by User

Names that consist either of a single lowercase letter or a lowercase letter followed by a character other than a lowercase letter (names .c2 and .nP are already used) should be used to avoid

duplication with already used names. The following is a possible naming convention:

macros:	aA (e.g., bG, kW)
strings:	as (e.g., c), f], p})
registers:	a (e.g., f, t)

O.2 Sample Extensions

O.2.1 Appendix Headings

The following is a way of generating and numbering appendix headings:

```
.nr Hu 1
.nr a 0
.de aH
.nr a +1
.nr P 0
.PH ''''Appendix \\na-%'''
.SK
.HU ''\\$1''
..
```

After the above initialization and definition, each call of the form

`.aH` title

begins a new page (with the page header changed to "Appendix *a-n*") and generates an unnumbered heading of *title*, which, if desired, can be saved for the table of contents. To center appendix titles the *Hc* register must be set to 1 {§D.2.2.3}.

O.2.2 Hanging Indent With Tabs.

The following example illustrates the use of the hanging indent feature of variable-item lists {§E.1.1.6}. In this example, a user-defined macro is defined to accept four arguments that make up the *mark*. In the output, each argument is to be separated from the previous one by a tab; tab settings are defined later. Since the first argument may begin with a period or apostrophe, the "\&" is used so that the formatter will not interpret such a line as a formatter request or macro call.

The 2-character sequence "`\&`" is understood by formatters to be a "zero-width" space. It causes no output characters to appear, but it removes the special meaning of a leading period or apostrophe.

The "`\t`" is translated by the formatter into a tab. The "`\c`" is used to concatenate the input text that follows the macro call to the line built by the macro. The user-defined macro and an example of its use are:

```
.de aX
.LI
\&\\$1\t\\$2\t\\$3\t\\$4\t\c"
..
.ta 8 14 20 24
.VL 24
.aX .nh off \- no
Automatic hyphenation is turned off."
Words containing hyphens"
may still be split across lines."
.aX .hy on \- no
Automatic hyphenation is turned on."
.aX .hc\<sp>c none none no
Hyphenation indicator is set to ‘‘c’’ or removed."
The indicator is suppressed such that it"
will not appear in the output."
Prepending the indicator to a word has the effect"
of preventing hyphenation of that word."
.LE
```

where <sp> stands for a space.

The resulting output is:

.nh	off	-	no	Automatic hyphenation is turned off. Words containing hyphens may still be split across lines.
.hy	on	-	no	Automatic hyphenation is turned on.
.hc c	none	none	no	Hyphenation indicator is set to “c” or removed. The indicator is suppressed such that it will not appear in the output. Prepending the indicator to a word has the effect of preventing hyphenation of that word.

P. Summary

The following are qualities of MM that have been emphasized in its design in approximate order of importance:

- *Robustness in the face of error* — A user need not be an*nr off*/*troff* expert to use MM macros. When the input is incorrect, either the macros attempt to make a reasonable interpretation of the error or an error message describing the error is produced. An effort has been made to minimize the possibility that a user would get cryptic system messages or strange output as a result of simple errors.
- *Ease of use for simple documents*— It is not necessary to write comple x sequences of commands to produce documents. Reasonable macro argument default values are provided where possible.
- *Parameterization* — There are many different preferences in the area of document styling. Many parameters are provided so that users can adapt input text files to produce output

documents to their respective needs over a wide range of styles.

- *Extension by moderately expert users* — A strong effort has been made to use mnemonic naming conventions and consistent techniques in construction of macros. Naming conventions are given so that a user can add new macros or redefine existing ones if necessary.
- *Device independence*— A common use of MM is to produce documents on hard cop y via teletypewriter terminals using the *nroff* formatter. Macros can be used conveniently with both 10- and 12-pitch terminals. In addition, output can be displayed on an appropriate CRT terminal. Macros have been constructed to allow compatibility with the *troff* formatter so that output can be produced on both a phototypesetter and a teletypewriter/CRT terminal.
- *Minimization of input*— The design of macros attempts to minimize repetiti ve typing. For example, if a user wants to have a blank line after all first- or second-level headings, the user need only set a specific parameter once at the beginning of a document rather than type a blank line after each such heading.
- *Decoupling of input format from output style*— There is b ut one way to prepare the input text although the user may obtain a number of output styles by setting a few global flags. For example, the .H macro is used for all numbered headings, yet the actual output style of these headings may be made to vary from document to document or within a single document.

Q. MM and Formatter Error Messages

When processing text using the MM macro package, MM will report any errors it can detect. Since the macros are written in the basic formatter primitives, other errors may be found and reported by the formatter (*nroff*, *troff*). The next two sections contain descriptions of the error messages which may be received from MM or the formatter.

Q.1 MM Error Messages

An MM error message has a standard part followed by a variable part. The standard part has the form:

ERROR:(*filename*)input line *n*:

Variable parts consist of a descriptive message usually beginning with a macro name. They are listed below in alphabetical order by macro name, each with a more complete explanation.

Check TL, AU, AS, AE, MT sequence

The correct order of macros at the start of a memorandum is shown in section {§F} You have not used the correct order, or one or more is missing.

Check TL, AU, AS, AE, NS, NE, MT sequence

The correct order of macros at the start of a memorandum is shown in section {§F} You have not used the correct order, or one or more is missing. (Occurs if the .AS 2 {§F.5} macro was used.)

Check WA, WE, IA, IE, LT sequence

The correct order of these macros is shown in {§G}. You have not used the correct order, or one or more is missing.

CS: cover sheet too long

Text of the cover sheet is too long to fit on one page. The abstract should be reduced or the indent of the abstract should be decreased {§F.5}.

DE: no DS or DF active

A .DE macro has been encountered, but there has not been a previous .DS or .DF macro to match it.

DF: illegal inside TL or AS

Displays are not allowed in the title or abstract.

DF: missing DE

A .DF macro occurs within a display, i.e., a .DE macro has been omitted or mistyped.

DF: missing FE

A display starts inside a footnote. The likely cause is the omission (or misspelling) of a .FE macro to end a previous footnote.

DF: too many displays

More than 26 floating displays are active at once, i.e., have been accumulated but not yet output.

DS: illegal inside TL or AS

Displays are not allowed in the title or abstract.

DS: missing DE

A .DS macro occurs within a display, i.e., a .DE has been omitted or mistyped.

DS: missing FE

A display starts inside a footnote. The likely cause is the omission (or misspelling) of a .FE to end a previous footnote.

FE: no FS active

A .FE macro has been encountered with no previous .FS to match it.

FS: missing DE

A footnote starts inside a display, i.e., a .DS or .DF occurs without a matching .DE.

FS: missing FE

A previous .FS macro was not matched by a closing .FE, i.e., an attempt is being made to begin a footnote inside another one.

H: bad arg:*value*

The first argument to the .H macro must be a single digit from one to seven, but *value* has been supplied instead.

H: missing arg

The .H macro needs at least one argument.

H: missing DE

A heading macro (.H or .HU) occurs inside a display.

H: missing FE

A heading macro (.H or .HU) occurs inside a footnote.

HU: missing arg

The .HU macro needs one argument.

LB: missing arg(s)

The .LB macro requires at least four arguments.

LB: too many nested lists

Another list was started when there were already six active lists.

LE: mismatched

The .LE macro has occurred without a previous .LB or other list- initialization macro {§E.1.1}. This is not a fatal error. The message is issued because there exists some problem in the preceding text.

LI: no lists active

The .LI macro occurred without a preceding list-initialization macro. The latter probably has been omitted or entered incorrectly.

LO: Required argument missing

The .LO macro requires an argument.

LO: argument not recognized

You have use an invalid argument with .LO.

LT: LT argument not recognized

You have use an invalid argument with .LT.

ML: missing arg

The .ML macro requires at least one argument.

ND: missing arg

The .ND macro requires one argument.

RF: no RS active

The .RF macro has been encountered with no previous .RS to match it.

RP: missing RF

A previous .RS macro was not matched by a closing .RF.

RS: missing RF

A previous .RS macro was not matched by a closing .RF.

S: bad arg:value

The incorrect argument *value* has been given for the .S macro {§M.10}.

SA: bad arg:value

The argument to the .SA macro (if any) must be either 0 or 1. The incorrect argument is shown as *value*.

SG: missing DE

The .SG macro occurred inside a display.

SG: missing FE

The .SG macro occurred inside a footnote.

SG: no authors

The .SG macro occurred without any previous .AU macro(s).

VL: missing arg

The .VL macro requires at least one argument.

)W: WA macro missing

You have not entered a .WA/.WE pair when .LT was used.

)W: WA or WE macro missing

If you use .WA you must specify .WE and visa-versa.

)W: WA, WE, or IE macro missing

You have omitted either or both of the .IA and .IE macros.

WC: unknown option

An incorrect argument has been given to the .WC macro {§M.5}.

Q.2 Formatter Error Messages

Most messages issued by the formatter are self-explanatory. Those error messages over which the user has some control are listed below. Any other error messages should be reported to the local system support group.

Cannot do ev

Caused by:

1. setting a page width that is negative or extremely short
2. setting a page length that is negative or extremely short
3. reprocessing a macro package (e.g., performing a `.so` request on a macro package that was already requested on the command line)
4. requesting the *troff* formatter −s1 option on a document that is longer than ten pages.

Cannot execute *filename*

Given by the **.!** request if the *filename* is not found.

Cannot open *filename*

Indicates one of the files in the list of files to be processed cannot be opened.

Exception word list full

Indicates too many words have been specified in the hyphenation exception list (via `.hw` requests).

Line overflow

Indicates output line being generated was too long for the formatter line buffer capacity. The excess was discarded. Likely causes for this message are very long lines or words generated through the misuse of `\c` of the `.cu` request, or very long equations produced by **eqn**/**neqn**.

Nonexistent font type

Indicates a request has been made to mount an unknown font.

Nonexistent macro file

Indicates the requested macro package does not exist.

Nonexistent terminal type

Indicates the terminal options refer to an unknown terminal type.

Out of temp file space

Indicates additional temporary space for macro definitions, diversions, etc. cannot be allocated. This message often occurs because of unclosed diversions (missing .FE or .DE), unclosed macro definitions (e.g., missing "**..**"), or a huge table of contents.

Too many page numbers

Indicates the list of pages specified to the **−o** formatter option is too long.

Too many number registers

Indicates the pool of number register names is full. Unneeded registers can be deleted by using the **.rr** request.

Too many string/macro names

Indicates the pool of string and macro names is full. Unneeded strings and macros can be deleted using the **.rm** request.

Word overflow

Indicates a word being generated exceeded the formatter word buffer capacity. Excess characters were discarded. Likely causes for this message are very long lines, words generated through the misuse of \c of the **.cu** request, or very long equations produced by **eqn**/**neqn**.

R. MM Macro Name Summary

The following list shows all the MM macros and their usage, a brief description, and the page number where more detailed information may be found.

Those macros marked with an asterisk (‘‘*’’) are not, in general, called directly by the user. These are ‘‘user exits’’ defined by the user and called by the MM macros from inside header, footer, or other macros.

Macro	*Description*	*Page*
.HC [indicator]	Hyphenation character	4-3
.P [type]	Paragraph	4-3
.nP	Numbered paragraph	4-4
.H level [text] [suffix]	Numbered heading	4-4
.HM [arg1 . . . arg7]	Heading mark	4-5
.HU text	Unnumbered heading	4-6
.HX dlevel rlevel heading-text	Pre-header macro	4-6
.HY dlevel rlevel heading-text	Pre-header macro	4-6
.HZ dlevel rlevel heading-text	Post-header macro	4-6
.AL [type] [text indent] [1]	Automatically numbered list	4-7
.BL [text indent] [1]	Bullet list	4-7
.DL [text indent] [1]	Dash list	4-7
.ML mark [text indent] [1]	Marked list	4-7
.RL [text indent] [1]	Reference list	4-8
.VL tindent [mindent] [1]	Variable-item list	4-8
.LI [mark] [1]	List item	4-8
.LE [1]	List end	4-8
.LB tin min pad typ [mk] [LIsp] [LBsp]	List begin	4-9
.LC [level]	List clear	4-10
.TL [charge #] [file #]	Memorandum title	4-10
.AU name [i] [loc] [dept] [x] [r] [a] [a]	Author information	4-10
.AT [title] . . .	Author’s title	4-10
.TM [number] . . .	Memorandum numbers	4-10
.AE	Abstract end	4-10
.AS	Abstract start	4-10
.OK [keyword] . . .	Other keywords	4-11
.MT [type] [addressee]	Memorandum type	4-11
.ND [new-date]	New date	4-11
.AF [company-name]	Alternate Subject/Date/From block	4-11
.FC [closing]	Formal closing	4-13
.SG [arg] [1]	Signature	4-13
.NS [arg] [1]	Notation start	4-13
.NE	Notation end	4-13
.AV name [1]	Approval signature	4-14
.LT [arg]	Letter type	4-14
.WA name [title]	Writer’s address	4-15
.WE	Writer’s address end	4-15
.IA	Inside address start	4-15
.IE	Inside address end	4-15
.LO CN	Letter option	4-15
.DS [format] [fill] [rindent]	Static display start	4-16
.DE	Static display end	4-16
.DF [format] [fill] [rindent]	Floating display start	4-17
.DE	Floating display end	4-17

.RD [prompt] [diversion] [string]	Interactive read	4-27

S. MM String Name Summary

The following list shows the predefined string names used by the MM macro package, a brief description of their use, and a page number where more detailed information may be found.

String	*Description*	*Page*
DT	Current date, unless overridden	4-1
BU	Bullet	4-3
EM	Em (long) dash	4-3
Tm	Trademark Symbol	4-3
HF	Header font list	4-5
HP	Header point size list	4-5
Lf	Title for List of Figures	4-18
Lt	Title for List of Tables	4-18
Lx	Title for List of Exhibits	4-18
Le	Title for List of Equations	4-18
F	Footnote number generator	4-18
Ci	Table of contents indent list	4-23
Rf	Reference number generator	4-23
Rp	Title for References	4-24
RE	Release ID of MM macros	4-24

If the released-paper style is used, then (in addition to the above strings) certain BTL location codes are defined as strings. These location strings are needed only until the .MT macro is called {§F.7}. Currently, the following codes are recognized:

AK, AL, ALF, CB, CH, CP, DR, FJ, HL, HO, HOH, HP, IH, IN, INH, IW, MH, MV, PY, RD, RR, WB, WH, and WV.

Section A.6 has notes on setting and referencing strings.

T. MM Number Register Summary

The list that follows contains a description of all the predefined number registers used by MM, a brief description of their use, and a page number where more detailed information may be found.

Any register having a single-character name can be set from the command line {§B.3}. An asterisk (*) attached to a register name indicates that this register can be set ***only*** from the command line or before the MM macro definitions are read by the formatter {§B.3}.

Section A.6 has notes on setting and referencing registers.

Register	*Description*	*Page*
D*	Debug flag	4-2
E*	Controls font of Subject/Date/From	4-2
L*	Length of page	4-2
N*	Numbering and header/footer style	4-2
O*	Offset of page	4-2
S*	Default point size	4-2
T*	Type of output device (nroff only)	4-2
U*	Underlining style for .H and .HU (nroff only)	4-2
W*	Width of page	4-2
Hy	Hyphenation control	4-3
Pi	Paragraph indent	4-4
Pt	Paragraph indentation type	4-4
Np	Numbering style for paragraphs	4-4
Ps	Spacing between paragraphs	4-4
Ej	Page-eject flag for headings	4-4
Hb	Heading break level after .H and .HU	4-4
Hs	Heading space level after .H and .HU	4-4
Hi	Heading temporary indent after .H and .HU	4-4
Hc	Heading centering level for .H and .HU	4-5
H1–H7	Heading counters, levels 1–7	4-5
Ht	Heading type for .H	4-6
Hu	Heading level for unnumbered headings (.HU)	4-6
Cl	Level of headings saved for Table of Contents	4-6
Li	List indent amount	4-7
Ls	Spacing between list items	4-7
Au	Inhibits printing of author information	4-10
A*	Set to omit Subject/Date/From block	4-11
Si	Standard indent for displays	4-17
Ds	Static display pre- and post-space	4-17
De	Display eject register for floating displays	4-17
Df	Display format register for floating displays	4-17
Eq	Equation label placement	4-18
Ec	Equation counter, used by .EC	4-18
Ex	Exhibit counter, used by .EX	4-18
Fg	Figure counter, used by .FG	4-18
Tb	Table counter, used by .TB	4-18
Of	Figure caption style	4-18
Lf	Flag to print List of Figures	4-18
Lt	Flag to print List of Tables	4-18
Le	Flag to print List of Equations	4-18
Lx	Flag to print List of Exhibits	4-18
Fs	Spacing between footnotes	4-19
C*	Copy type (Original, DRAFT, etc.)	4-21
P	Page number	4-21
Pv	PRIVATE heading	4-22
Cp	Placement of List of Figures	4-23
Oc	Table of contents page numbering style	4-23
Rf	Reference counter used by .RS	4-23