MM — A Macro Package for Generating Documents

1. INTRODUCTION

This document is the user's guide and reference manual for the Memorandum Macros (MM), a general-purpose package of text formatting macros for use with the UNIX® system text formatters *troff* and *nroff*.

The user must be familiar with the UNIX system and with the use of an editor. Some familiarity with the request summary in the *Troff User's Manual* is helpful. *A Troff Tutorial* provides a general overview of formatter capabilities, while *Troff User's Manual* provides detailed information as well as a summary of requests.

1.1 Purpose

The purpose of MM is to provide a unified, consistent, and flexible tool for producing many common types of documents. Although the UNIX system provides other macro packages for various specialized formats, MM has become the standard, general-purpose macro package for most documents.

MM can be used to produce:

- Letters
- Reports
- Technical Memoranda
- Released Papers
- Manuals
- Books

The uses of MM range from single-page letters to documents of several hundred pages in length, such as user guides, design proposals, etc.

1.2 Conventions

Each section of this memorandum explains a single facility of MM. In general, the earlier a section occurs, the more necessary it is for most users. Some of the later sections can be completely ignored if MM defaults are acceptable. Likewise, each section progresses from normal-case to special-case facilities. Read a section in detail only until there is enough information to obtain the desired format, then skim the rest of it because some details may be of use to just a few people. Sections that require knowledge of the formatters have a bullet (•) at the end of their headings.

A constant-width typeface is used for all instances of computer input. In all examples, keyboard input is shown in `constant-width`, and macro arguments and text representing user information are shown in the standard *italic* font. In this text, macro, number register and string names are in constant-width (e.g., `.P`, `.H`, `Pi`, `HF`).

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.

Many macros generate blank lines or vertical space. In this text, each reference to ‘‘blank vertical space’’ means a half vertical line. Note that two blank vertical spaces means one full vertical line (two half-lines).

1.3 Overall Structure of a Document

The input for a document that is to be formatted with MM possesses four major segments, any of which may be omitted; if present, they *must* occur in the following order:

- *Parameter-setting* — This segment sets the general style and appearance of a document. The user can control page width, margin justification, numbering styles for headings and lists, page headers and footers, and many other properties of the document. Also, the user can add macros or redefine existing ones. This segment can be omitted entirely if one is satisfied with default values; it produces no actual output, but only performs the setup for the rest of the document.
- *Beginning* — This segment includes those items that occur only once, at the beginning of a document, e.g., title, author's names, date.
- *Body* — This segment is the actual text of the document. It may be as small as a single paragraph, or as large as hundreds of pages. It may have a hierarchy of *headings* up to seven levels deep. Headings are automatically numbered (if desired) and can be saved to generate the table of contents. Five additional levels of subordination are provided by a set of *list* macros for automatic numbering, alphabetic sequencing, and ''marking'' of list items. The body may also contain various types of displays, tables, figures, references, and footnotes.
- *Ending* — This segment contains those items that occur once only, at the end of a document. Included here are signatures and notation lists (e.g., ''copy to'' lists). Certain macros may be invoked here to print information that is wholly or partially derived from the rest of the document, such as the table of contents or the cover sheet for a document.

The existence and size of these four segments varies widely among different document types. Although a specific item (such as date, title, author's name, etc.) may be printed in several different ways depending on the document type, there is a uniform way of typing it in.

1.4 Definitions

The term *formatter* refers to either of the text-formatting programs *troff* and *nroff*. The examples of output in this manual are as produced by *troff*; *nroff* output would, of course, look somewhat different. The discussion of commands and features focusses on the behavior of MM with *troff*; in some cases, *nroff* behavior is different. For example: where *troff* italicizes, *nroff* underlines; *troff* values are always scaled (e.g., `1.5i`, `3n`, `5p`), *nroff* values are unscaled (e.g., `15`, `3`, `5`); *nroff* ignores point size and vertical spacing requests. Appendix A provides a complete summary of those *nroff* actions that differ from *troff*.

Requests are built-in commands recognized by the formatter. Although one seldom needs to use these requests directly, this document contains references to some of them. For example, the request:

```
.sp
```

inserts a full blank line in the output.

Macros are named collections of requests. Each macro is an abbreviation for a collection of requests that would otherwise require repetition. MM 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.

Strings provide character variables, each of which names a string of characters. Strings are often used in page headers, page footers, and lists. They share the pool of names used by *requests* and *macros*. 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 one-character names) or ''`\*(`'' (for two-character names). For instance, the string `DT` in MM normally contains the current date, so that the input line:

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

may result in the following output:

Today is April 1, 1990.

The current date can be replaced, e.g.:

```
.ds DT 01/01/90
```

or by invoking a macro designed for that purpose.

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

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

See §14.1 regarding naming conventions for requests, macros, strings, and number registers. Appendix H lists all macros, strings, and number registers defined in MM.

2. INVOKING THE MACROS

This section tells how to access MM and shows the typical UNIX command lines to format an MM document.

2.1 The –mm Flag

The MM package can be invoked 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 any other files. This action defines the MM macros, sets default values for various parameters, and initializes the formatter to be ready to process the files of input text.

2.2 Typical Command Lines

The prototype command lines are as follows (with the various options explained in §2.3 and in the *Troff User's Manual*).

- Plain text (no tables, equations, pictures or graphs):

  `troff -mm` [*options*] *filename* | …

- Text with tables:

  `tbl` *filename ...* | `troff -mm` [*options*] | …

- Text with equations:

  `eqn usr/pub/eqnchar` *filename* | `troff -mm` [*options*] | …

- Text with both tables and equations:

  `tbl` *filename* | `eqn /usr/pub/eqnchar -` | `troff -mm` [*options*] | …

- Text with pictures:

  `pic` *filename* | `troff -mm` [*options*] | …

- Text with graphs:

  `grap` *filename* | `pic` | `troff -mm` [*options*] | …

- Text with the works (tables, equations, graphs and pictures):

  `grap` *filename* | `pic` | `tbl` | `eqn` | `troff -mm` [*options*] | …

Preprocessing by *tbl* and *eqn*, if needed, must be invoked as shown in the command line prototypes. If used, *eqn* immediately precedes *troff*. Other preprocessors (*tbl*, *pic*, *grap*) are used before *eqn* and *troff*, but need not be in any particular order. Since *grap* is a preprocessor to *pic*, you must use *grap* before *pic*.

2.3 Parameters that Can Be Set from the Command Line

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

`−rA`*n* modifies the first-page format for memoranda and letters. If *n* is 1 (or any non-zero value), the letterhead block is suppressed to accommodate preprinted stationary. The letterhead block includes the company name and logo as well as any other signature element that appears above the ‘‘subject/date/from’’ block.

`−rC`*n* sets the type of copy (e.g., DRAFT) to be printed at the bottom of each page:

n	Type of Copy
0	none (default)
1	OFFICIAL FILE COPY
2	DATE FILE COPY
3	DRAFT with single-spacing and default paragraph style
4	DRAFT with double-spacing and 10-en paragraph indent
5	double-spacing with 10-en paragraph indent

`−rD1` sets *debug mode*. This flag requests the formatter to attempt to continue processing even if MM detects errors that would otherwise cause termination. It also includes some debugging information in the default page header.

`−rE`*n* controls the font of the subject/date/from fields on memoranda and letters. If *n* is 1, then these fields are emboldened. If *n* is 0, then these fields appear in the normal text font.

`−rL`*k* sets the length of the physical page to *k* where *k* is a scaled value. The default page length is 11 inches (e.g., `11i`).

`−rN`*n* specifies the page numbering style.

n	Page 1	Pages 2ff
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	same as 3, with ‘‘*section-figure*’’	same as page 1

The contents of the prevailing header and footer do not depend on the value of the number register `N`; `N` only controls whether and where the header (and the footer if `N` is 3 or 5) is printed, as well as the page numbering style. In particular, if the header and footer are null, the value of `N` is irrelevant.

`−rO`*k* sets the page offset (e.g., left margin) to *k* where *k* is a scaled value. The default page offset is 1 inch (e.g., `1i`). (The register name is the capital letter O, not the digit zero 0.)

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

`−rS`*n* sets the point size and vertical spacing for the document. By default, *n* is 10, i.e., 10-point type on 12-point vertical spacing.

`−rW`*k* sets the page width (i.e., line length and title length) to *k* where *k* is a scaled value. The default page width is 6 inches (e.g., `6i`).

2.4 Omission of –mm

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:

```
any initializations of registers in §2.3
.so /usr/lib/tmac/tmac.m
remainder of text
```

In this case, one must not use the –mm flag; the `.so` request has the equivalent effect, but the registers in §2.3 must be initialized before the `.so` request, because their values are meaningful only if set before the 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 L 9i
.nr W 5i
.nr O 0.5i
.nr N 3
.so /usr/lib/tmac/tmac.m
.H 1 "INTRODUCTION"
 …
```

specifies a page length of 9 inches, a page width (i.e., line length) of 5 inches, a page offset of one-half inch, and ‘‘section-page’’ numbering.

3. FORMATTING CONCEPTS

3.1 Basic Terms

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

Certain formatting commands (requests and macros) cause the filling of the current output line to cease, the line (of whatever length) to be printed, and the 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.

While formatter requests can be used with MM, one must fully understand the consequences and side-effects that each such request might have. Actually, there is little need to use formatter requests; the macros described here 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 or tables of contents) can be obtained with ease.
- the user is insulated from the peculiarities of the formatter language.

A good rule is to use formatter requests only when absolutely necessary.

In order to make it easy to revise the input text at a later time, input lines should be kept short and should be broken at the end of clauses; each new full sentence must begin on a new line.

3.2 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 `""`. Note that omitting an argument is not the same as supplying a null argument (for example, see the `.MT` macro). Furthermore, omitted arguments can occur only at the end of an argument list, while null arguments can occur anywhere.

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 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 you must, use two grave accents (`‘‘`) and/or two acute accents (`’’`) 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 (re)printed in the table of contents.

3.3 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 harm the desired alignment of text. To avoid this problem, it is necessary to be able to specify a space that cannot be expanded during justification, i.e., an ‘‘unpaddable space.’’ There are several ways to accomplish this.

First, one may use the built-in *troff* escape sequence ‘‘\□’’ where □ is a single space from the keyboard. This sequence directly generates an unpaddable space. Second, one may sacrifice some seldom-used character to be translated into space upon output. Because this translation occurs after justification, the chosen character may be used anywhere an unpaddable space is desired. The tilde (˜) is often used for this purpose. To use it in this way, insert the following at the beginning of the document:

```
.tr ˜
```

You can then use ˜ in the text to represent an unpaddable space. If a tilde must actually appear in the output, it can be temporarily ‘‘recovered’’ by inserting:

```
.tr ˜˜
```

before the place where it is needed (this translates ˜ back into itself). Its previous usage is restored by repeating the ‘‘`.tr ˜`’’, but only after a break or after the line containing the tilde has been forced out.

3.4 Hyphenation

The formatters do not perform hyphenation unless the user requests it. Hyphenation can be turned on in the body of the text by specifying:

```
.nr Hy 1
```

once at the beginning of the document.

If hyphenation is requested, the formatters will automatically hyphenate words, if need be. However, the user may specify the hyphenation points for a small list of words or for a specific occurrence of any word by the use of a special character known as a ‘‘hyphenation indicator.’’

If the hyphenation indicator (initially, the two-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. In any case, all occurrences of the hyphenation indicator disappear on output.

The user may specify a different hyphenation indicator:

`.HC` [*hyphenation-indicator*]

The circumflex ˆ is often used for this purpose; this is done by inserting the following at the beginning of a document:

```
.HC ˆ
```

Any word containing hyphens or dashes — also known as *em* 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 supply, via the `.hw` request, a small list of words with the proper hyphenation points indicated. For example, to indicate the proper hyphenation of the word ‘‘printout,’’ one may specify:

```
.hw print-out
```

3.5 Tabs

The macros `.MT`, `.TC`, and `.CS` use the formatter `.ta` request to set tab stops, and then restore the default values of tab settings. By default, tabs are set every one-half inch.

Note that a tab character is always 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’’ mode. Also note that *tbl* changes tab stops, but does not restore the default tab settings.

3.6 Special Use of the BEL Character

The non-printing character BEL is used as a delimiter in many macros where it is necessary to compute the width of an argument or to delimit arbitrary text, e.g., in headers and footers, headings, and list marks. Users who include BEL characters in their input text (especially in arguments to macros) will receive mangled output.

3.7 Strings for Special Symbols

3.7.1 Bullets. The string `\*(BU` produces a bullet (•) in the proper size and vertical position on the line for use in text. Note that the bullet list macro `.BL` uses this string to generate the bullets for its list items.

3.7.2 Dashes, Minus Signs, and Hyphens. *Troff* has distinct graphics for a dash (—), a minus sign (–), and a hyphen (-):

Dash	Use the string `\*(EM` for a text dash. This type of dash—commonly called the em dash—is used as a mark of separation (as shown here). Note that the dash list macro `.DL` generates the em dash for its list items.
Hyphen	Use the character `-` to produce a hyphen.
Minus	Use the sequence `\-` for a true minus sign.

3.7.3 Registered and Unregistered Marks. Use the string `\*(Rg` to produce the registration ® symbol for registered marks. For unregistered marks, use the strings `\*(Tm` and `\*(Sm` to produce the trademark ™ and the service mark ℠ symbols, respectively. Marks should always be used as adjectives when they appear in printed text (e.g., UNIX® system, Tuxedo™ software, ACCUNET℠ digital services).

3.8 Use of Formatter Requests •

Most formatter requests should not be used with MM because MM provides equivalent functionaltiy in a much more user-oriented and surprise-free fashion than do the basic formatter requests. However, some formatter requests are useful with MM, namely:

```
.af   .br   .ce   .de   .ds   .fi   .hw   .ls   .nf   .nr
.nx   .rm   .rr   .rs   .so   .sp   .ta   .ti   .tl   .tr
```

The `.fp`, `.lg`, and `.ss` requests are also sometimes useful. Use of other requests without fully understanding their implications very often leads to disaster.

4. PARAGRAPHS AND HEADINGS

This section describes simple paragraphs and section headings. Additional paragraph styles are covered later in this section; list styles are covered in the next section.

4.1 Paragraphs

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

This macro is used to begin two kinds of paragraphs. In a ‘‘blocked’’ (e.g., left-justified) paragraph, the first line begins at the left margin, while in an ‘‘indented’’ paragraph, the first line is indented 3 ens.

A document possesses a default paragraph style obtained by specifying `.P` before each paragraph that does not follow a heading. The default style is controlled by the register `Pt`. The initial value of `Pt` is 0, which always provides blocked paragraphs. To force all paragraphs to be indented, insert:

```
.nr Pt 1
```

at the beginning of the document. To indent all paragraphs except those that follow headings, lists, and displays, insert:

```
.nr Pt 2
```

at the beginning of the document.

The amount of paragraph indentation is contained in the register `Pi`; this amount must be an unscaled number and is treated as ens. By default, `Pi` is set to 3. To indent paragraphs by 10 ens, insert:

```
.nr Pi 10
```

at the beginning of the document. Of course, both the `Pi` and `Pt` register values must be greater than zero for any paragraphs to be indented.

The number register `Ps` controls the amount of spacing between paragraphs. By default, `Ps` is set to 1, yielding one blank vertical space.

Regardless of `Pt`, an individual paragraph can be forced to be blocked or indented. Use `.P` with *type* `0` (e.g., ‘‘`.P 0`’’) to force left justification; *type* `1` (e.g., ‘‘`.P 1`’’) always causes indentation by the amount specified by the register `Pi`. If `.P` occurs inside a list (see §5), the indent (if any) of the paragraph is added to the current list indent.

Numbered paragraphs may be produced by setting the register `Np` to 1. This produces paragraphs numbered within first level headings (e.g., 1.01, 1.02, 1.03, 2.01, etc.). A different style of numbered paragraphs is obtained by using the `.nP` macro rather than the `.P` macro for paragraphs. This produces paragraphs that are numbered within second level headings and contain a ‘‘double-line indent’’ in which the text of the second line is indented to be aligned with the text of the first line so that the number stands out.

```
.H 1 "FIRST HEADING"
.H 2 "Second Heading"
.nP
one or more lines of text
```

4.2 Numbered Headings

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

The `.H` macro provides seven levels of numbered headings, as illustrated by this document. Level 1 is the most major or highest; level 7 the lowest.

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

Strictly speaking, there is no need for a `.P` macro immediately after a `.H` (or `.HU`), because the `.H` macro also performs the function of the `.P` macro, and an immediately following `.P` 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.

4.2.1 Normal Appearance. The effect of `.H` varies according to the *level* argument. First-level headings are preceded by two blank vertical spaces; all others are preceded by one blank vertical space.

`.H 1` *heading-text*	gives a bold heading followed by one blank vertical space. The following text begins on a new line and is indented according to the current paragraph type. Normally, full capital letters are used to make the heading stand out.
`.H 2` *heading-text*	yields a bold heading followed by one blank vertical space. The following text begins on a new line and is indented according to the current paragraph type. Normally, initial capitals are used.
`.H` *n heading-text*	for $3 \leq n \leq 7$, produces an italic heading followed by two spaces. The following text appears on the same line, i.e., these are run-in headings.

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

Here are the `.H` calls of §4:

```
.H 1 "PARAGRAPHS AND HEADINGS"
.H 2 "Paragraphs"
.H 3 "Numbered Headings"
.H 3 "Normal Appearance."
.H 3 "Altering Appearance of Headings."
.H 4 "Pre-Spacing and Page Ejection."
.H 4 "Spacing After Headings."
.H 4 "Centered Headings."
.H 4 "Heading Fonts."
.H 4 "Heading Point Sizes."
.H 4 "Marking Styles\*(EMNumerals and Concatenation"
.H 2 "Unnumbered Headings"
.H 2 "Headings and the Table of Contents"
.H 2 "First-Level Headings and Page Numbering Style"
.H 2 "User Exit Macros \*(BU"
.H 2 "Hints for Large Documents"
```

4.2.2 Altering Appearance of Headings. Users satisfied with the default appearance of headings may skip to §4.3. One can modify the appearance of headings quite easily by setting certain registers and strings at the beginning of the document. 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.

4.2.2.1 Pre-Spacing and Page Ejection. A first-level heading normally has two blank vertical spaces preceding it, and all others have one blank vertical space. If a multi-line 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. Long documents may be made more manageable if each section starts on a new page. Setting `Ej` 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 `Ej`.

4.2.2.2 Spacing After Headings. Three registers control the appearance of text immediately following a `.H` call. They 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 occurs after the heading. If the heading level is less than or equal to `Hs`, a blank vertical space is inserted after the heading. Defaults for `Hb` and `Hs` are 2. If a heading level is greater than `Hb` and also greater than `Hs`, then the heading (if any) is run into the following text. 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.

For any stand-alone heading, i.e., a heading not run into the following text, the alignment of the next line of output is controlled by the register `Hi`. If `Hi` is 0, text is left-justified. If `Hi` is 1 (the default value), the text is indented according to the paragraph type as specified by the register `Pt`. Finally, 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.

For example, to cause a blank 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 `Pt`, the following should appear at the top of the document:

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

4.2.2.3 Centered Headings. The register `Hc` can be used to obtain centered headings. A heading is centered if its level is less than or equal to `Hc`, and if it is also stand-alone. For example,

```
.nr Hs 2
.nr Hc 2
```

centers first- and second-level headings. By default, headings are not centered; register `Hc` is 0 initially.

4.2.2.4 Heading Fonts. The string `HF` contains seven codes that specify the fonts for heading levels 1 through 7. The digits `1`, `2`, and `3` are used to select the roman, italic, and bold font, respectively. By default, the `HF` string is defined as

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

Thus, the first two levels are bold and the remaining levels are italic.

The user may reset `HF` as desired. Any value omitted from the end of the list is taken to be 1. For example,

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

would result in three bold levels, two italic levels, and two roman (normal text font) levels.

4.2.2.5 Heading Point Sizes. The user may also specify the desired point size for each heading level with the `HP` string. By default, the text of headings (`.H` and `.HU`) is printed in the same point size as the body except that bold stand-alone headings are printed in a size 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. Each value can be an absoluate or a relative point size. Null or zero values imply that the default size will be used for the corresponding heading level. 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. On the other hand, given relative point sizes changes:

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

specifies that the first- and second-level headings are two points larger, the third- and fourth-level headings are one point smaller, and the remaining level headings appear in the default size.

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

Note: Only the point size of the heading is affected. Specifying a large point size without providing increased vertical spacing (via `.HX` or `.HZ`) may cause overprinting.

4.2.2.6 Marking Styles — Numerals and Concatenation

`.HM` [*style1*] ... [*style7*]

The registers names `H1` through `H7` are used as counters for the seven levels of headings. Their 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 *style* arguments, and each argument is a string indicating the type of marking to be used:

style	Interpretation
`1`	Arabic (default for all levels)
`0001`	Arabic with enough leading zeroes to get the specified number of digits
`A`	Upper-case alphabetic
`a`	Lower-case alphabetic
`I`	Upper-case Roman
`i`	Lower-case Roman

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, set the register `Ht` (heading-mark type) to 1. For example, a commonly-used outline style is obtained by:

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

4.3 Unnumbered Headings

`.HU` *heading-text*

`.HU` 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 2` *heading-text*

is the printing of the heading mark for the latter. Both 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.

`.HU` 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.

4.4 Headings and the 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 two things:

- specifying in the register `Cl` what level headings are to be saved
- invoking the `.TC` macro (see §10.1) at the end of the document

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

Due to the way the headings are saved, it is possible to exceed the formatter’s storage capacity, particularly when saving many levels of many headings, while also processing displays and footnotes. If this happens, the ‘‘Out of temp file space’’ diagnostic (see Appendix F) will be issued; the only remedy is to save fewer levels and/or to have fewer words in the heading text.

4.5 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 form ‘‘*section-page*,’’ where *section* is the number of the current first-level heading and *page* is the number of the page in that section. This page numbering style can be achieved by specifying the `−rN3` or `−rN5` flag on the command line. As a side effect, this also has the effect of setting `Ej` to 1, i.e., each 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 used.

4.6 User Exit 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. MM does not define `.HX`, `.HY` and `.HZ`; they are intended to be defined by the user. The `.H` macro invokes `.HX` shortly before the actual heading text is printed; it calls `.HZ` as its last action. After `.HX` is invoked, 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 invoked so that the user may respecify these features. All the default actions occur if these macros are not defined. If the `.HX`, `.HY` or `.HZ` are defined by the user, the user-suppled definition is interpreted at the appropriate point. These macros can therefore influence the handling of all headings, because the `.HU` macro is actually a special case of the `.H` macro.

If the user originally invoked the `.H` macro, then the derived level (*dlevel*) and the real level (*rlevel*) are both equal to the level given in the `.H` invocation. If the user originally invoked the `.HU` macro, *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, produced blank vertical space to precede the heading, 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 non-zero, 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. A value of 0 means that the heading is run-in. A value of 1 means a break (but no blank vertical space) is to follow the heading. A value of 2 means that a blank 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 non-zero, 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 vertical spaces) plus the contents of register `;3` (taken as blank vertical spaces) plus the number of lines of the heading.

The user may alter the values of `}0`, `}2` and `;3` within `.HX` as desired. The following are examples of actions that might be performed by defining `.HX` to include the lines shown (\□ represents an unpaddable space):

- Change first-level heading mark from format ‘‘*n*`.`’’ to ‘‘*n*`.0`’’:

```
    .if \\$1=1 .ds }0 \\n(H1.0\□\□
```

- Separate run-in heading from the text with a period and two unpaddable spaces:

  ```
  .if \\n(;0=0 .ds }2 .\□\□
  ```

- Assure that at least 15 lines are left on the page before printing a first-level heading:

  ```
  .if \\$1=1 .nr ;3 15−\\n(;0
  ```

- Add 3 additional blank lines before each first-level heading:

  ```
  .if \\$1=1 .sp 3
  ```

- Indent level 3 run-in headings by 5 ens:

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

If temporary strings or macros are used within `.HX`, choose their names with care (see §14.1).

`.HY` is called after the `.ne` is issued. Certain features requested in `.HX` must be repeated. For example:

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

`.HZ` is called at the end of `.H` to permit user-controlled actions after the heading is produced. For example, 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''"
..
```

4.7 Hints for Large Documents

A large document is often organized for convenience into one 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 1 less than the number of the section just before the corresponding `.H 1` call. For example, at the beginning of section 5, insert:

```
.nr H1 4
```

Note: This is a dangerous practice — it defeats the automatic (re)numbering of sections when sections are added or deleted. Remove such lines as soon as possible.

5. LISTS

This section describes many different kinds of lists: automatically-numbered and alphabetized lists, bullet lists, dash lists, lists with arbitrary marks, and lists starting with arbitrary strings.

5.1 Basic Approach

In order to avoid repetitive typing of arguments to describe the appearance of items in a list, MM provides a convenient way to specify lists. All lists are composed of the following parts:

- A *list-initialization* macro that controls the appearance of the list: line spacing, indentation, marking with special symbols, and numbering or alphabetizing.
- One or more *list item* (`.LI`) macros, each followed by the actual text of the corresponding list item.
- The *list end* (`.LE`) macro that terminates the list and restores the previous indentation.

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

With this approach, the format of a list is specified only once at the beginning of that list. In addition, by building on the existing structure, users may create their own customized sets of list macros with relatively little effort (see §5.4 and Appendix B).

Spacing at the beginning of the list and between the items can be suppressed by setting the `Ls` (list space) register. `Ls` 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 6 (which is the maximum list nesting level).

5.2 Sample Nested Lists

The input for several lists is shown below and the corresponding output is shown on the following page. The `.AL` and `.DL` macro calls contained therein are examples of the list-initialization macros. This example will help us to explain the material in the following sections.

```
.AL A
.LI
This is an alphabetized item.
This text shows the alignment of the second line of the item.
The quick brown fox jumped over the lazy dog's back.
.AL
.LI
This is a numbered item.
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 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 appears at the left margin.
```

Sample Nested Lists — Input

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

   1. This is a numbered 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. 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 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 appears at the left margin.

Sample Nested Lists — Output

5.3 Basic List Macros

Because all lists share the same overall structure except for the list-initialization macro, we first discuss the macros common to all lists. Each list-initialization macro is covered in §5.3.3.

5.3.1 List Item

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

The `.LI` macro is used with all lists. It normally causes the output of one blank vertical space before its item, although this may be suppressed. If no arguments are given, it labels its item with the current mark which is specified by the most recent list-initialization macro. If a single argument is given to `.LI`, 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
But this uses ''plus'' as prefix to the bullet.
.LE
```

yields:

- This is a simple bullet item.

+ This replaces the bullet with a ''plus.''

+ • But this uses ''plus'' as prefix to the bullet.

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 ''outdented,'' starting at the same place where the mark would have started.

Note: The mark must not contain ordinary (paddable) spaces, because alignment of items will be lost if the right margin is justified.

5.3.2 List End

`.LE [1]`

The list end macro restores the state of the list back to that existing just before the most recent list-initialization macro call. If the second argument is the digit `1`, then the list is followed by one blank 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 a blank vertical space of its own, such as `.P`, `.H` or `.LI`.

`.H` and `.HU` automatically clear all list information, so one may legally omit the `.LE`(s) that would normally occur just before either of these macros. Such a practice is *not* recommended, however, because errors will occur if the list text is separated from the heading at some later time (e.g., by insertion of text).

5.3.3 List Initialization Macros. The following are the various list-initialization macros. They are actually implemented as calls to the more basic `.LB` macro (see §5.4).

Each type of list has a default indentation appropriate for that style. It is possible to override the default indentation for a particular list by supplying the *text-indent* argument to the list initialization macro. If a value is given for the *text-indent* argument, that value must be an unscaled number and is treated as ens; if *text-indent* is null, the default indentation is used.

The spacing between list items may be suppressed for a particular list. If the last argument to the list initialization macro is the digit `1`, a blank vertical space will not separate the items in the list. However, a blank vertical space will occur before the first item.

5.3.3.1 Automatically-Numbered or Alphabetized Lists

`.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 5 ens from the indent in force when `.AL` is called, thus leaving room for a space, two digits, a period, and two spaces before the text.

The *type* argument may be given to obtain a different type of sequencing, and its value should indicate the first element in the sequence desired, i.e., it must be `1`, `A`, `a`, `I` or `i`. If *type* is omitted or null, then 1 is assumed.

By default, the indentation for an automatically-numbered list is taken from the value in register `Li`, which is initially 5 ens. Register `Li` may be changed to specify a different indent for all automatically-numbered lists. The *text-indent* argument overrides the default indentation for that list only.

5.3.3.2 Bullet List

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

The `.BL` macro begins a bullet list in which each item is marked by a bullet (•) followed by one space. By default, the indentation for a bullet list is the same as for an indented paragraph as given in register `Pi`. The *text-indent* argument overrides the default indentation for that list only.

5.3.3.3 Dash List

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

The `.DL` macro is identical to `.BL`, except that an em dash (—) is used instead of a bullet.

5.3.3.4 Marked List

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

The `.ML` macro is much like `.BL` and `.DL`, but expects the user to specify an arbitrary mark which may consist of more than a single character. By default, text is indented one more space than the width of *mark*. The *text-indent* argument overrides the default indentation for that list only.

5.3.3.5 Reference List

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

The `.RL` macro begins an automatically-numbered reference list in which the numbers are enclosed by square brackets [ ]. By default, the indentation for a reference list is 5 ens, a convenient value for lists numbered up to 99. The *text-indent* argument overrides the default indentation for that list only.

5.3.3.6 Variable-Item List

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

When a list begins with a `.VL`, 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. The *mark-indent* argument gives the number of spaces from the current indent to the beginning of the mark; *mark-indent* defaults to 0 if omitted or null. The *text-indent* argument gives the distance (in ens) from the current indent to the beginning of the text. For example:

```
.VL 20 2
.LI "mark\ 1"
Here is a description of mark 1;
the argument ``mark 1'' of the .LI line contains
an unpaddable space in order to avoid extra spaces
between ``mark'' and ``1''.
.LI "second\ mark"
This is the second mark, also using 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 that follows.
.LI "\ "
This item effectively has no mark due to the unpaddable space
following the .LI macro.
The unpaddable space is needed for this list item,
otherwise a ``hanging indent'' would have been produced.
.LE
```

yields:

mark 1	Here is a description of mark 1; the argument ‘‘mark 1’’ of the .LI line contains an unpaddable space in order to avoid extra spaces between ‘‘mark’’ and ‘‘1’’.
second mark	This is the second mark, also using 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 that follows.
	This item effectively has no mark due to the unpaddable space following the .LI macro. The unpaddable space is needed for this list item, 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 ens.
.LE
```

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 ens.

5.4 List-Begin Macro and Customized Lists •

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

The list-initialization macros described above suffice for almost all cases. However, one may obtain more control over the layout of lists by using the basic list-begin macro `.LB`, which is also used by all the other list-initialization macros. (Appendix B shows how to use `.LB` to build other kinds of lists.)

The argument *text-indent* gives the number of ens that text is indented from the current indent. Normally, this value is taken from register `Li` for automatic lists and from register `Pi` for bullet and dash lists.

The combination of *mark-indent* and *pad* 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). Within the mark area, the mark is left-justified if *pad* is 0. If *pad* is greater than 0, say *n*, blanks are appended to the mark; the *mark-indent* value (typically 0) is ignored. The resulting string immediately precedes the text. That is, the mark is effectively right-justified *pad* spaces immediately to the left of the text.

The arguments *type* and *mark* interact with one another to control the marking style. If *type* is 0, simple marking is performed using the format given in *mark*. If *type* is greater than 0, automatic numbering or alphabetizing is performed and *mark* is interpreted as the first item in the sequence for numbering or alphabetizing (chosen from the set `1`, `A`, `a`, `I`, `i` as in §5.3.3.1).

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

Each non-zero value of *type* from 1 to 6 selects a different way of displaying the marks:

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

where *x* is the generated number or letter.

LI-space gives the number of blank vertical spaces 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. *LB-space* gives the number of blank vertical spaces to be output by `.LB` itself. If omitted, *LB-space* defaults to 0.

There are three reasonable 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 vertical space before each item in the list; such a list is usually terminated with a `.LE 1` to end the list with a blank vertical space. In the second case, for a more compact list, set *LI-space* to 0 and *LB-space* to 1, and, again, use `.LE 1` at the end of the list. The result is a list with one blank vertical space before and after it. If you set both *LI-space* and *LB-space* to 0, and use `.LE` to end the list, a list without any blank vertical spaces will result.

6. DOCUMENT STYLES

The MM package provides several document styles: memoranda, released papers, and external letters that conform to a given company standard, and common business letters in standard acceptable styles.

6.1 Memoranda, Released Papers, and External Letters

One use of MM is for the preparation of memoranda and released papers, which have special requirements for the first page and for the cover sheet. The information needed for the memorandum or released paper (title, author, date, etc.) is entered in the same way for both styles; an argument to one macro (`.MT`) indicates which style is being used. The following sections describe the macros used to provide this data. The required order is shown in §6.1.9.

The same macros for memorandum and released paper styles are used to generate external letters on preprinted stationary with company letterhead. Appendix C shows the input and output samples for the three document styles: memoranda, released paper, and external letter.

If no particular document style is desired, the macros described in this section should be omitted from the input text. If these macros are omitted, the first page will simply have the page header followed by the body of the document.

6.1.1 Title

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

The arguments to the `.TL` macro are the charging case number(s) and filing case number(s). The ‘‘charging case’’ is the case number to which time was charged for the development of the project described in the memorandum (also referred to as Work Project Number). The ‘‘filing case’’ is a number under which the memorandum is to be filed (e.g., for library reference).

The title of the memorandum or paper follows the `.TL` macro and is processed in fill mode. 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. Multiple filing case numbers are entered similarly. For example:

```
.TL "12345, 67890" 987654321
On the Construction of a Table
of All Even Prime Numbers
```

The `.br` request may be used to break the title on output into several lines as follows:

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

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

If only a charging case number or only a filing case number is given, then 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 ‘‘Work Project Number and Filing Case’’ followed by the number will appear on a line following the title. If the two case numbers are different, then separate lines for ‘‘Work Project Number’’ and ‘‘File Case’’ will appear after the title.

6.1.2 Author(s)

`.AU` *name* [*initials*] [*location*] [*organization*] [*phone*] [*room*] [*arg*] [*arg*] [*arg*]
`.AT` [*title*] …

The `.AU` macro receives as arguments information that describes an author. If any argument contains blanks it 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 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 HO 98760 "(201) 949-1234" 1Z-234
.AT Director "Materials Research Laboratory"
```

In the ‘‘from’’ portion in the memorandum style, the author's name is followed by ‘‘Org.’’ and the organization number on one line, the location and room number together on one line, and the phone number* on another line. The printing of the location, organization number, room number and phone 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. The last three *arg* arguments 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 miscellaneous information, such as the author's electronic address. For example:

```
.AU "S. P. Lename" SPL MH 98765 "(201) 582-1234" 2G-444 attmail!spl
```

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

The names of the authors in the released-paper style are centered below the title. After the name of the last author, the company name and the location are centered. For the case of authors from different locations, see §6.1.8; for authors from different companies, see §6.1.7.2.

6.1.3 Document Number(s)

`.TM` [*number*] …

Internal documents (e.g., technical memorandum or internal memorandum) are usually identified by document numbers. The document identification numbers are supplied via the `.TM` macro:

```
.TM 12345-890101-01TM 12355-890101-03TM
```

Up to nine numbers may be specified. On output, the document identification number(s) will be listed following the ‘‘from’’ section of memoranda.

6.1.4 Abstract

`.AS` [*mode*] [*indent*] [*type*]
one or more lines of
of abstract text
`.AE`

The `.AS` (abstract start) and `.AE` (abstract end) macros bracket the optional abstract. Abstracts are printed on the first page of a document and/or on its cover sheet (if any). Headings and displays are not permitted within an abstract.

* The argument *phone* should be the complete phone number, including area code.

In a released paper and in a technical memorandum, if the *mode* argument is 0, the abstract will be printed on page 1 and on the cover sheet (if any); if the *mode* argument is 1, the abstract will appear only on the cover sheet (if any).

In an internal memorandum and in all other documents (other than external letters), if the *mode* argument is 0, the abstract will appear on the first page of the document and there will be no cover sheet printed; if the *mode* argument is 2, the abstract will appear only on the cover sheet, which will be produced automatically in this case (i.e., without invoking the `.CS` macro). It is not possible to get either an abstract or a cover sheet with an external letter.

The abstract is printed with ordinary text margins. If a value is given for the *indent* argument, it will be used as the amount of indentation from both the left and right margins. The indent amount must be an unscaled number and is treated as ens.

By default, the abstract heading is ''ABSTRACT''. If the *type* argument is `ER`, the abstract heading is ''ERRATA''; if the *type* argument is `AD`, the abstract heading is ''ADDENDUM''.

Headings and displays are not permitted within an abstract.

6.1.5 Other Keywords

`.OK` [*keyword*] …

Up to nine topical keywords or keyword phrases may be specified as arguments to the `.OK` macro. If a keyword argument contains spaces, it must be enclosed within double quotes.

6.1.6 Memorandum Types

`.MT` [*type*] [*address*]

The `.MT` macro controls the format of the top part of the first page of a memorandum or of a released paper, as well as the format of the cover sheet (if any). The *type* argument is used as follows to produced the corresponding document styles:

type	Document Style	Memorandum Heading
`""`	Memorandum	
`0`	Memorandum	
none	Memorandum	TECHNICAL MEMORANDUM
`1`	Memorandum	TECHNICAL MEMORANDUM
`2`	Memorandum	INTERNAL MEMORANDUM
`3`	Memorandum	ADMINISTRATIVE MEMORANDUM
`4`	Released Paper	
`5`	External Letter	
"*string*"	Memorandum	*string*

If *type* indicates a memorandum style, then the appropriate heading 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 memorandum is produced by calling `.MT` with a null (but not omitted) or zero argument.

The second argument to `.MT` is the name of the addressee of a memorandum or external letter; if present, that name and the page number replaces the normal page header on the second and succeeding pages. For example:

```
.MT 1 "John Jones"
```

produces:

John Jones – 2

With the released-paper style (`.MT 4`), the second argument serves a different purpose (see §6.1.8).

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

6.1.7 Date and Format Changes

6.1.7.1 Changing the Date. By default, the current date appears in the ‘‘date’’ part of a memorandum. This can be overridden by using:

`.ND` *new-date*

The `.ND` macro alters the value of the string `DT`, which is initially set to the current date.

6.1.7.2 Alternate First-Page Format. One can specify an alternate format for the first page of memoranda and change the company name used for any document style.

`.AF` [*company-name*]

If an argument is given, *company-name* replaces the default company name on the first page of a released paper and in the letterhead block of a memoranda. If the argument is null, the company name is suppressed; in this case, an extra blank line is inserted to allow room for stamping the document with an appropriate name or signature element.

For memoranda, use the `-rA`*n* option (see §2.3) to supress the entire letterhead block to accommodate preprinted stationary. The letterhead block includes the company name and logo as well as other signature elements (i.e., rule across the page between logo and name) that appear before the ‘‘subject/date/from’’ block.

The command line option `–rE`*n* controls the font of the information fields in the ‘‘subject/date/from’’ block on memoranda.

The user-definable string `}Z` contains the company name, and the string `]S` contains the company logo. These string definitions are initialized in the file `/usr/lib/tmac/strings.mm`. (Use the `.AF` macro to redefine the company name string.)

6.1.8 Released-Paper Style. The released-paper style is obtained by specifying:

`.MT 4 [1]`

The released paper style provides a titlebox that contains:

- the title of the paper (emboldened)
- the author’s names (italicized), each on a separate line
- the company name
- the company address (city, state, zip code)

Each line in the titlebox is centered. The location of the last author is used to determine the company address. If the second argument is the digit `1`, then the name of each author is followed by the respective company name and address.

Information necessary for the memorandum style but not for the released-paper style is ignored. In addition, the macros described in §6.3 and their associated lines of input are ignored when the released-paper style is specified.

In the released-paper style, location codes are defined as strings that contain the address of the corresponding location. (See Appendix G for a list of location codes and addresses.) Authors from non-affiliated locations may include their company name in the released-paper style by specifying the company name with the `.AF` macro and defining a string (with a two-character name such as `XX`) for the address before each `.AU`.

For example:

```
.TL
A Learned Treatise
.AU "Samuel P. Lename" SPL CB
.AF "Getem Electronics, Inc."
.ds XX "Philadelphia, Pennsylvania  19103
.AU "Frank G. Swatter" FGS XX
.MT 4 1
```

produces:

A Learned Treatise

Samuel P. Lename
General Ceramics, Inc.
Columbus, Ohio 43213

Frank G. Swatter
Getem Electronics, Inc.
Philadelphia, Pennsylvania 19103

(Note that in this example, ''General Ceramics, Inc.'' is the default company name.)

6.1.9 Sequence of ''Beginning'' Memorandum/Paper/Letter Macros. The macros described in §6.1, if present, must be given in the following order:

`.ND` *new-date*
`.TL` [*charging-case*] [*filing-case*]
one or more lines of title text
`.AF` [*company-name*]
`.AU` *name* [*initials*] [*location*] [*organization*] [*phone*] [*room*] [*arg*] [*arg*] [*arg*]
`.AT` [*title*] ...
`.TM` [*number*] ...
`.AS` [*mode*] [*indent*] [*type*]
one or more lines of abstract text
`.AE`
`.MT` [*type*] [*addressee*]
`.P`
body of document
 ...
`.SG` [*initials*] `[1]`
`.NS` [*type*] `[1]`
lines of notation list
`.NE`

The only required macros for a memorandum or a released paper are `.TL`, `.AU`, and `.MT`; all the others (and their associated input lines) may be omitted if the features they provide are not needed. All macros before `.MT` simply collect information for their respective parts. The `.MT` macro then uses that information to produce the document in the desired format. (The macros following `.MT` are discussed in §6.3.)

6.2 Business Letters

In addition to the document styles produced with `.MT`, you can obtain an entirely different style using a set of macros designed to produce common business letters. The information needed for business letters is entered in the same way; an argument to one macro (`.LT`) indicates which style is being used. Appendix D shows the input and output samples for each of the four types of common business letters.

6.2.1 Letter Types

`.LT` [*type*]

The letter-type macro formats a business letter in one of four common styles: blocked, semi-blocked, full-blocked, and simplified. Use the *type* argument to select the desired format:

type	Letter Format
`BL`	***Blocked*** — The date, return address, closing and writer's identification lines begin at the center of the page. All other text begins at the left margin.
`SB`	***Semi-blocked*** — The same as the blocked format, except the first line of each paragraph is indented 3 ens.
`FB`	***Full-blocked*** — All lines begin at the left margin.
`SP`	***Simplified*** — The same as the full-blocked format, except the salutation is replaced by a subject line and is followed by an additional blank line, the closing is omitted, and the writer's identification is on a single line. The subject and writer's identification lines are in all-capital letters.

If *type* is omitted, `.LT` produces the ``blocked'' letter format.

6.2.2 Writer's Address

`.WA` *name* [*title*]
one or more lines
of the writer's address
`.WE`

The writer's complete address (but not the name) appears in the return-address block at the top of the letter. The writer's name and title, if any, appear in the signature block at the close of the letter.

6.2.3 Recipient's Address

`.IA` [*name* [*title*]]
[*name*]
one or more lines
of the recipient's address
`.IE`

The inside address appears at the left margin before the salutation, if any. The inside address contains the recipient's complete address and may also include the recipient's name and title. If the *name* and *title* arguments are given, they are used in the inside address, each on a separate line, before the recipient's address; otherwise, a name may be given as part of the address.

6.2.4 Letter Options

`.LO` *type* [*notation*]

The `.LO` (letter options) macro provides the salutation and four additional components frequently used in business correspondence. Use the *type* argument to select the desired component:

type	Component
`SA`	Salutation
`AT`	Attention line
`SJ`	Subject line
`RN`	Reference line
`CN`	Confidential notation

A business letter may contain one or more component, or none at all. Thus, the `.LO` macro may be ignored or it may be used multiple times in any order to generate different components.

6.2.4.1 Salutation. If *type* is `SA`, the `.LO` macro generates the salutation given in *notation*. If *notation* is not given, the saluation ‘‘To Whom It May Concern:’’ is used.

The salutation appears two blank vertical spaces below the inside address at the left margin.

6.2.4.2 Attention Line. If *type* is `AT`, the `.LO` macro generates an attention line of the form:

ATTENTION: *notation*

If *notation* is not given, the attention field is blank (e.g., only ‘‘ATTENTION:’’ is used).

The attention line appears two blank vertical spaces below the inside address at the left margin. The attention line is used to direct the letter to the attention of a specific person or department.

6.2.4.3 Subject Line. If *type* is `SJ`, the `.LO` macro generates a subject line of the form:

Subject: *notation*

If *notation* is not given, the subject field is blank (e.g., only ‘‘Subject:’’ is used).

The subject line appears one blank vertical space below the salutation at the left margin. Using a subject line allows your correspondents (or their secretary) to locate previous correspondence on the same subject.

6.2.4.4 Reference Line. If *type* is `RF`, the `.LO` macro generates a reference line of the form:

In reference to: *notation*

If *notation* is not given, the reference field is blank (e.g., only ‘‘In reference to:’’ is used).

The reference line appears two blank vertical spaces below the date and is aligned left with the date. Using a reference line allows your correspondents (or their secretary) to locate previous correspondence on the same subject.

6.2.4.5 Confidential Notation. If *type* is `CN`, the `.LO` macro generates the notation ‘‘Confidential’’ below the date, aligned left with the date. If *notation* is given, it replaces ‘‘Confidential’’ below the date.

6.2.5 Sequence of Business Letter Macros The macros `.WA`, `.WE`, `.IA`, `.IE`, and `.LT` are mandatory and must be given in the following order:

```
.ND new-date
.WA name [title]
lines of writer’s address
.WE
.IA [name [title]]
[name]
lines of recipient’s address
.IE
.LO type [notation]
.LT [type]
.P
body of letter
 ...
.FC [closing]
.SG [initials] [1]
.NS [type] [1]
lines of notation list
.NE
```

If the `.ND` (see §6.1.7.1) or `.LO` macros are used, they must appear before the `.LT` macro. All macros before `.LT` simply collect information for their respective parts. The `.LT` macro then uses that information to produce the letter in the desired format. (The macros following `.LT` are discussed in the next section.)

6.3 Macros for the End of a Memorandum or Letter

The signatures of the authors and a list of notations can be requested at the end of a memorandum, external letter, or business letter. The following macros provide information for the signature block and notation list; these macros and their input are ignored for the released-paper style. (See the examples in Appendix C and Appendix D.)

6.3.1 Signature Block. The signature block consists of the formal closing (if any), a signature line for each author, and an optional reference line. The signature line includes the author's name preceded by a sufficient amount of space for the author's signature.

The signature block is not used in the released-paper style.

6.3.1.1 Formal Closing

`.FC` [*closing*]

The `.FC` macro generates the formal closing at the end of an external letter. The closing appears before the signature, so the `.FC` macro must precede the `.SG` macro. By default, ‘‘Yours very truly,’’ is used as the formal closing. If the *closing* argument is given, it will be used as the closing. For example,

```
.FC "Sincerely,"
```

generates ‘‘Sincerely,’’ as the formal closing before the signature.

6.3.1.2 Signature and Reference Line

`.SG` [*initials*] [`1`]

The `.SG` macro prints the author's names 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.

A reference line appears at the left margin on the same line as the last author's name. The reference line contains the location code, organization number, author's initials, and typist's initials, separated by hyphens. The typist's initials are supplied as an argument to the `.SG` macro; the location code, organization number, and author's initials are taken from the `.AU` macro. If the `.SG` macro is used with no arguments, the reference line is suppressed. If the *initials* argument is null, the reference line is printed without the typists's initials nor the preceding hyphen.

If there are several authors and if the second argument is the digit `1`, then the reference data is placed on the same line as the name of the first author, rather than the last author. In this case, the reference data contains only the location and organization number of the first author. Thus, if there are authors from different organizations and/or from different locations, the reference data should be supplied manually after the `.SG` macro:

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

In this case, the `.SG` macro must be used alone, with no arguments.

6.3.2 ‘‘Copy to’’ and Other Notations

`.NS` [*type*] [`1`]
zero or more lines
of notation list
`.NE`

After the signature and reference data, many types of notations may follow, such as a list of attachments or ‘‘copy to’’ lists. The various notations are obtained through the `.NS` macro, which provides for the proper spacing and for breaking the notations across pages, if necessary.

The codes for *type* and the corresponding notations are:

type	Notations
0	Copy to
1	Copy (with att.) to
2	Copy (without att.) to
3	Att.
4	Atts.
5	Enc.
6	Encs.
7	U.S.C.
8	Letter to
9	Memorandum to
10	Copy (with atts.) to
11	Copy (without atts.) to
12	Abstract Only to
13	Complete memorandum to
14	Cover Sheet Only to
"*string*"	Copy (*string*) to
"*string*" 1	*string*

If *type* is omitted or null, the default notation ‘‘Copy to’’ is used. If *type* consists of a multiple-character text string, 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. If the second argument is the digit 1, then *string* is used as the entire notation.

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

```
.NS 4
Attachment 1\*(EMList of register names
Attachment 2\*(EMList 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

The .NS and .NE macros may also follow .AS 2 and .AE to place the notation list on the cover page, if any. If notations are given at the beginning without .AS 2, they will be saved and output at the end of the document.

6.3.3 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. The approver's name is printed below the signature line. For example:

```
.AV "R. U. Sure"
```

produces:

APPROVED:

____________________________________ __________________

R. U. Sure Date

If the second argument is the digit `1`, then the ''APPROVED:'' mark is suppressed. This feature is helpful for successive calls to `.AV` when multiple approval lines are required. For example:

```
.AV "R. U. Sure"
.AV "I. M. Certain" 1
.AV "B. A. Sporte" 1
```

produces:

APPROVED:

____________________________________ __________________

R. U. Sure Date

____________________________________ __________________

I. M. Certain Date

____________________________________ __________________

B. A. Sporte Date

6.4 Forcing a One-Page Letter

At times, one would like just a bit more space on the page, forcing the signature or items within notations onto the bottom of the page, so that the letter or memo is just one page in length. This can be accomplished by increasing the page length through the `−rL`*n* option (e.g., `−rL12i` for a 12-inch page length). This has the effect of making the formatter believe that the page is longer and therefore has more room than usual to place the signature or the notations. This will only work for a single-page letter or memorandum.

7. DISPLAYS AND CAPTIONS

Displays are blocks of text that are to be kept together — not split across pages. MM provides two styles of displays: a ''static'' style (`.DS`) and a ''floating'' style (`.DF`). 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 big to fit there. In the floating style, the display ''floats'' through the input text to the top of the next page if there is not enough room for it on the current page; thus the 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 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 indentation or centering, as well as fill-mode processing.

Displays and footnotes may never be nested in any combination. Although lists and paragraphs are permitted, no headings (`.H` or `.HU`) can occur within displays or footnotes.

7.1 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` will accept the lines of text exactly as they are typed (no-fill mode) and will not indent them from the prevailing left margin indentation or from the right margin. The *rindent* argument is the amount by which the line length should be decreased, i.e., an indentation from the right margin. (This amount may be scaled or else defaults to ems.)

The *format* argument to `.DS` is an integer or letter used to control the left margin indentation and centering with the following meanings:

format	Meaning
`0` or `L`	left-justified (default)
`1` or `I`	indent by standard amount
`2` or `C`	center each line
`3` or `CB`	center as a block

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

fill	Meaning
`0` or `N`	no-fill mode (default)
`1` or `F`	fill mode

The standard amount of indentation is taken from the register `Si`, which is initially 3 ens. Thus, by default, the text of an indented display aligns with the first line of indented paragraphs, whose indent is contained in the `Pi` register. Even though their initial values are the same, these two registers are independent of one another. (The value for registers `Si` and `Pi` must be an unscaled number and are treated as ens.)

The display format value `3` (`CB`) centers the entire display as a block (as opposed to `.DS C` and `.DF 2`, which center each line individually). That is, all the collected lines are left-justified, and then the display is centered, based on the width of the longest line. This format must be used in order for the *eqn* ‘‘mark’’ and ‘‘lineup’’ feature to work with centered equations (see §7.4). This format must be used to center pictures and graphs as well (see §7.5 and §7.6).

By default, a blank vertical space is placed before and after static and floating displays. These blank vertical spaces before and after static displays can be inhibited by setting the register `Ds` to 0.

The following example uses all three arguments to display a block of text that is filled and indented 5 ens from both the left and right margins. The input:

```
.nr Si 5
.DS I F 5n
‘‘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
```

yields:

> ‘‘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.’’

7.2 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. The arguments have the same meanings as for `.DS`, except that, for floating displays, indent, no indent, and centering are always calculated with respect to the initial left margin, because the prevailing indent may change between the time when the formatter first reads the floating display and the time that the display is printed. One blank vertical space always occurs both before and after a floating display.

The user may exercise great control over the output positioning of floating displays through the use of two number registers, `De` and `Df`. When a floating display is encountered, it is processed and placed onto a queue of displays waiting to be output. Displays are always removed from the queue and printed in the order that they were entered on the queue, which is the order that they appeared in the input file. If a new floating display is encountered and the queue of displays is empty, then the new display is a candidate for immediate output on the current page. Immediate output is governed by the size of the display and the setting of the `Df` register. The `De` register controls whether or not text will appear on the current page after a floating display has been produced.

`Df`	Action
`0`	Floating displays will not be output until the end of the section (with ‘‘section-page’’ numbering) or the end of the document.
`1`	Output the new floating display on the current page if there is room, otherwise hold it until the end of the section or document.
`2`	Output one floating display from the queue at the top of a new page or column (when in two-column mode).
`3`	Output one floating display on the current page if there is room, otherwise output one floating display at 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. Note that if register `De` 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. (This also applies to value 5 below.)
`5`	Output a new floating display on the current page if there is room. Output a least one, but as many displays as will fit starting at the top of a new page or column. (This is the default action.)
	Note: If `Df` is set to any other value, the action performed is the same as for value 5.

`De`	Action
`0`	No special action occurs. (This is the default action.)
`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.
	Note: If `De` is set to any other value, the action performed is the same as for value 1.

As long as the 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 two-column mode) the next display from the queue becomes a candidate for output if the `Df` register has specified ‘‘top-of-page’’ output. When a display is output it is also removed from the queue. When the end of a section (with section-page numbering) or the end of a document is reached, all displays are automatically output. This occurs before a `.SG`, `.CS`, or `.TC` macro is processed.

A display is said to ‘‘fit on the current page’’ if there is enough room to contain the entire display on the page, or if the display is longer than one page in length and less than half of the current page has been used. In two-column mode, a wide (full page width) display will never fit in the second column. The `.WC` macro (see §12.3.1) may be used to control handling of displays in double-column mode and to control the break in the text before floating displays.

7.3 Tables with *tbl*

```
.TS [H]
global options ;
column descriptors .
header data
[.TH [N]]
data entries
.TE
```

The `.TS` (table start) and `.TE` (table end) macros are used to delimit the text to be examined by the *tbl* preprocessor as well as to set proper spacing around the table. The display function and the *tbl* delimiting function are independent of one another. However, a static or floating display should be used to keep together any mixture of tables, equations, filled and unfilled text, and captions.

The macros `.TS` and `.TE` also permit the processing of tables that extend over several pages. If a table heading is needed for each page of a multiple-page table, specify the argument `H` to the `.TS` macro as above. Following the options and format information, the table heading is typed on as many lines as required and followed by the `.TH` macro. The `.TH` macro must occur when `.TS H` is used. Note that this is not a feature of *tbl*, but of the macro definitions provided by MM.

The table header macro `.TH` 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 .
header data
.TH
data entries
.TE
.TS H
global options ;
column descriptors .
header data
.TH N
data entries
.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 (for example, too many blocks of filled text). If each segment had its own `.TS H/.TH` sequence, each segment would have its own header. However, if each table segment after the first uses `.TS H/.TH N` then the table header will appear at the beginning of the table and the top of each new page or column that the table continues onto.

7.4 Equations with *eqn*

```
.DS [format]
.EQ [label]
equation input
.EN
.DE
```

The `.EQ` (equation start) and `.EN` (equation end) macros are used to delimit the text to be examined by the *eqn* preprocessor. The `.EQ` and `.EN` macros must occur inside a display. There is an exception to this rule: if `.EQ` and `.EN` are used only to specify the delimiters for in-line equations or to specify *eqn* ‘‘defines’’, `.DS` and `.DE` must not be used; otherwise extra blank lines will appear in the output. In order for the *eqn* ‘‘mark’’ and ‘‘lineup’’ feature to work with centered equations, the entire set of equations must be enclosed in a centered-block display (display format `CB` or `3`).

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. If the register `Eq` is set to 1, the label will appear at the left margin. The equation will be centered for centered displays; otherwise the equation will be adjusted to the opposite margin from the label.

7.5 Pictures with *pic*

```
.DS [format]
.PS
picture input
.PE
.DE
```

The `.PS` (picture start) and `.PE` (picture end) macros are used to delimit the text to be examined by the *pic* preprocessor. The `.PS` and `.PE` macros must occur inside a display. A picture may be displayed at the left margin, indented, or centered as a block. Do not use a centered-line display (display format `C` or `2`) to center a picture; use a centered-block display instead (display format `CB` or `3`).

7.6 Graphs with *grap*

```
.DS [format]
.G1
graph input
.G2
.DE
```

The `.G1` (graph start) and `.G2` (graph end) macros are used to delimit the text to be examined by the *grap* preprocessor. The `.G1` and `.G2` macros must occur inside a display. A graph may be displayed at the left margin, indented, or centered as a block. Do not use a centered-line display (display format `C` or `2`) to center a graph; use a centered-block display instead (display format `CB` or `3`).

7.7 Figure, Table, Equation, and Exhibit Captions

```
.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 a display to automatically number and title figures, tables, equations, and exhibits, respectively. For example:

```
.FG "This is an illustration"
```

yields:

Figure 1. This is an illustration

The `.FG` macro generates ‘‘Figure’’ as the caption label; the `.TB`, `.EC`, and `.EX` macros generate ‘‘TABLE’’, ‘‘Equation’’, and ‘‘Exhibit’’ as their respective caption labels. Figures, tables, equations, and exhibits are automatically numbered using registers `Fg`, `Tb`, `Ec`, and `Ex`, respectively. The caption is centered if it can fit on a single line; otherwise, the second and succeeding lines of the caption title are indented after the caption label and number (as in a variable list).

The *override* string is used to modify the normal numbering. If *flag* is omitted or 0, *override* is used as a prefix to the number; if *flag* is 1, *override* is used as a suffix to the number; if *flag* is 2, *override* replaces the number. The option `−rN5` automatically sets ‘‘section-figure’’ numbering and *override* is ignored.

The register `Of` controls the mark that separates the caption label from the caption title. By default, `Of` is 0 and the separation mark consists of a period and two spaces (e.g., ‘‘**Figure 1.** Title’’). If `Of` is set to 1, the separation mark consists of a hyphen surrounded by spaces (e.g., ‘‘**Figure 1** - Title’’).

As a matter of style, table headings are usually placed ahead of the text of the tables, while figure, equation, and exhibit captions usually occur after the corresponding figures and equations.

7.8 List of Figures, Tables, Equations, and Exhibits

A list of figures, tables, exhibits, and equations will be printed after the Table of Contents if registers `Lf`, `Lt`, `Lx`, and `Le`, respectively, are set to 1. By default, `Lf`, `Lt`, and `Lx` are 1, and `Le` is 0. If any of the List of Figures, List of Tables, List of Equations, and List of Exhibits are printed, each list appears on a separate page following the Table of Contents. If register `Cp` is set to 1, each list is printed as part of the Table of Contents.

The title for the list of figures, tables, exhibits, and equations are contained in the strings `Lf`, `Lt`, `Lx`, and `Le`, respectively. By default, the strings are defined as:

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

To change the title for a given list, simply redefine the appropriate string.

8. FOOTNOTES

There are two macros that delimit the text of footnotes, a string used to automatically number the footnotes, and a macro that specifies the style of the footnote text.

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

The segment of text to be footnoted is immediately followed by the footnote mark. If a label is supplied to `.FS`, then it will be used to mark the footnote; otherwise the footnote will be numbered automatically. Automatically-numbered footnotes and user-labeled footnotes may be intermixed.

The `.FS` (footnote start) and `.FE` (footnote end) macros mark the beginning and end of the footnote text. The text between `.FS` and `.FE` is processed in fill mode. Another `.FS`, a `.DS` or a `.DF` are not permitted between the `.FS` and `.FE` macros.

Appendix E illustrates footnote styles using both automatically-numbered footnotes and labeled footnotes.

8.1 Automatically-Numbered Footnotes

Footnotes may be numbered automatically by 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. The footnote text follows, enclosed in the `.FS` and `.FE` macro pair. Since the footnote number is generated automatically, `.FS` is used with no argument. For example:

```
This is the segment of text\*F
.FS
This is the text of the footnote.
The same number is used to mark the text
and the footnote.
.FE
to be footnoted.
```

produces an automatically-numbered footnote.

The number register `:p` keeps track of the current footnote number. The string `F` automatically increments register `:p` and then uses that value as the footnote number in the footnote mark; the `.FS` macro uses that same value to mark the text of the footnote. The `:p` register may have its format or value changed to affect the footnote marks.

Automatically-numbered footnotes may not be used in a title (following `.TL`), in an abstract (between `.AS` and `.AE`), or in a table (between `.TS` and `.TE`).

8.2 Labeled Footnotes

Footnotes may be labeled manually by using the footnote label immediately after the text to be footnoted, without any intervening spaces. The footnote text follows, enclosed in the `.FS` and `.FE` macro pair. The same label used in the text must be supplied as an argument to the `.FS` macro. For example:

```
This is the segment of text*
.FS *
This is the text of the footnote.
An asterisk is used to mark the text
and to label the footnote.
.FE
to be footnoted.
```

produces a labeled footnote. If more than one labeled footnote appears on the same page, care must be taken to choose a different label for each footnote.

8.3 Spacing Between Footnote Entries

Normally, one blank vertical space separates the footnotes when more than one occurs on a page. To change this spacing, set the register `Fs` to the desired value. For example:

```
.nr Fs 2
```

will cause two blank vertical spaces to occur between footnotes.

8.4 Format of Footnote Text •

`.FD` [*mode*] [1]

The user can control the formatting style of the footnote text by specifying text hyphenation, right margin justification, and text indentation, as well as left- or right-justification of the label when text is indented. The `.FD` macro is used to select the style. The argument *mode* determines the particular combination of hyphenation, indentation, and margin and label justification that defines each style:

		Right Margin		Label
mode	Hyphenation	Justification	Text Indent	Justification
`0`	off	on	on	left-justified
`1`	on	on	on	left-justified
`2`	off	off	on	left-justified
`3`	on	off	on	left-justified
`4`	off	on	off	left-justified
`5`	on	on	off	left-justified
`6`	off	off	off	left-justified
`7`	on	off	off	left-justified
`8`	off	on	on	right-justified
`9`	on	on	on	right-justified
`10`	off	off	on	right-justified
`11`	on	off	on	right-justified

The default value for *mode* is 0. If *mode* is omitted, null, or out of range, the default value is used. (Hyphenation is turned on and off with the `.hy` and `.nh` requests, respectively. Right margin justification is turned on and off with the `.ad` and `.na` requests, respectively.)

If the second argument is the digit `1`, then automatically-numbered footnotes begin afresh with ‘‘1’’ when a first-level heading is encountered. This feature is most useful with ‘‘section-page’’ numbering.

Footnotes are separated from the body of the text by a short rule. Footnotes that continue to the next page are separated from the body of the text by a full-width rule. Footnotes are set in type that is two points smaller than the point size used in the body of the text.

For long footnotes that continue onto the following page, it is possible that, if hyphenation is permitted, the last line of the footnote on the current page will be hyphenated. Except for this case (which can be controlled by specifying an even-numbered value for *mode*), hyphenation across pages is inhibited by MM.

9. PAGE HEADERS AND FOOTERS

Text that occurs at the top of each page is known as the page header. Text printed at the bottom of each page is called the page footer. There can be up to three lines of text associated with the page header and the page footer: 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. Likewise, the page footer may have up to two lines of text: the line that occurs at the bottom of every page and the line for the even- or odd-numbered page.

This section describes the default appearance of page headers and page footers and ways to change them. The term ‘‘header’’ (not qualified by ‘‘even’’ or ‘‘odd’’) refers to the line of the page header that occurs on every page, and similarly for the term ‘‘footer’’.

9.1 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 when pages are numbered sequentially within sections. In a document whose style is controlled by the `.MT` or the `.LT` macro, the page header on the first page is automatically suppressed provided a break does not occur before the `.MT` or the `.LT` macro is called. The macros and text of §6.1-§6.2. and §9 as well as `.nr` and `.ds` requests do not cause a break and are permitted before the `.MT` or the `.LT` macro call.

Pages can be numbered sequentially within sections. To obtain this numbering style, specify `−rN3` or `−rN5` on the command line. In this case, the default footer is a centered *section-page* number (e.g., ‘‘9-1’’), and the default page header is blank.

The current page number is often used in the page header or the page footer (see §9.4). By default, the page number contained in the `P` register is an Arabic number. The format of the page number may be changed with the `.af` request (e.g., use format `i` for lower-case Roman numerals.)

If ‘‘debug mode’’ is set using the flag `−rD1` on the command line, additional information is included in the default header at the top left of each page.

9.2 Page Header

`.PH` [*header*]

The argument *header* is of the form

"`'`*left*`'`*center*`'`*right*`'`"

This form applies to the argument *header* for the `.EH` and `.OH` macros as well. If it is inconvenient to use the apostrophe (`'`) as the delimiter (i.e., because it occurs within one of the parts), it may be replaced uniformly by any other character. On output, the parts are left-justified, centered, and right-justified, respectively.

The `.PH` macro specifies the header that is to appear at the top of every page. By default, the page header is defined as a centered page number surrounded by hyphens.

9.2.1 Even-Page Header

`.EH` [*header*]

The `.EH` macro supplies a line to be printed at the top of each even-numbered page, immediately following the header. The even-page header is not defined by default so the initial value is a blank line.

9.2.2 Odd-Page Header

`.OH` [*header*]

The `.OH` macro supplies a line to be printed at the top of each odd-numbered page, immediately following the header. The odd-page header is not defined by default so the initial value is a blank line.

9.3 Page Footer

`.PF` [*footer*]

The argument *footer* is of the form

"`'`*left*`'`*center*`'`*right*`'`"

(same as for page headers). This form applies to the argument *footer* for the `.EF` and `.OF` macros as well. If it is inconvenient to use the apostrophe (`'`) as the delimiter (i.e., because it occurs within one of the parts), it may be replaced uniformly by any other character. On output, the parts are left-justified, centered, and right-justified, respectively.

The `.PF` macro specifies the line that is to appear at the bottom of each page. The page footer is not defined by default so the initial value is a blank line. If the `−rC`*n* flag is specified on the command line, the type of copy follows the footer on a separate line. In particular, if `−rC3` or `−rC4` (DRAFT) is specified, then, in addition, the footer is initialized to contain the date instead of being a blank line.

9.3.1 Even-Page Footer

`.EF` [*footer*]

The `.EF` macro supplies a line to be printed at the bottom of each even-numbered page, immediately preceding the footer. The even-page footer is not defined by default so the initial value is a blank line.

9.3.2 Odd-Page Footer

`.OF` [*footer*]

The `.OF` macro supplies a line to be printed at the bottom of each odd-numbered page, immediately preceding the footer. The odd-page footer is not defined by default so the initial value is a blank line.

9.3.3 Footer on the First Page. By default, page footers are not defined so the resulting lines are blank. If one specifies `.PF` and/or `.OF` before the end of the first page of the document, then 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` option is specified on the command line (see §2.3).

9.4 Use of Strings and Registers in Header and Footer Macros

String and register names may be placed in the arguments to the header and footer macros. If the value of the string or register is to be computed when the respective header or footer is printed, the invocation must be escaped by four (4) backlashes. This is because the string or register invocation will be processed three times:

- as the argument to the header or footer macro;
- in a formatting request within the header or footer macro;
- in a `.tl` request during header or footer processing.

For example, the page number register `P` must be escaped with four backslashes in order to specify a header in which the page number is to be printed at the right margin, e.g.:

```
.PH "'''Page \\\\nP'"
```

creates a right-justified header containing the word ‘‘Page’’ followed by the page number. Similarly, to specify a centered footer with the ‘‘section-page’’ style, use:

```
.PF "''- \\\\n(H1-\\\\nP -''"
```

As another example, suppose that 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 then be:

```
.PF "''\\\\*(a]''"
```

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

The following sequence specifies blank lines for the header and footer lines, page numbers on the outside edge 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:

```
.PH ""
.PF ""
.EH "'\\\\nP''Revision 3'"
.OH "'Revision 3''\\\\nP'"
```

9.5 Generalized Top-of-Page Processing •

During header processing, MM invokes two user-definable macros: the `.TP` macro and the `.PX` macro. The `.TP` macro is invoked in the environment (see `.ev` request) of the header; the `.PX` macro is a user-exit macro that is invoked (without arguments) when the normal environment has been restored, and with ‘‘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 (for every page), the string }e contains the even-page header, and the string }o contains the odd-page header, as defined by .PH, .EH and .OH, respectively. To obtain more specialized page titles, the user may redefine .TP to cause any desired header processing (see §12.3.2). For example, to obtain a page header that includes three centered lines of data (a document's number, issue date, and revision date), one could define .TP as follows:

```
.de TP
.sp
.ce 3
777—888—999
Iss. 2, AUG 1977
Rev. SEP 1977
.sp
..
```

The .PX macro may be used to supply 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.

9.6 Generalized Bottom-of-Page Processing

.BS
zero or more lines
of bottom-block 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. The bottom block will not appear on the cover sheet, if any. This block of text is removed by specifying an empty block:

```
.BS
.BE
```

9.7 Proprietary Markings and Copyright Notice

.PM [*type*] [*year*]

The .PM macro generates a proprietary marking or copyright notice. The marking appears at the bottom of the page following the page footer (if any).

By default, a document does not carry a proprietary marking or copyright notice. Use the .PM macro with the following value for *type* to generate the corresponding type of marking or notice:

type	Marking Type
none or 0	turn off previous marking, if any
P or 1	PROPRIETARY
RS or 2	PROPRIETARY (RESTRICTED)
RG or 3	PROPRIETARY (REGISTERED)
CP or 4	''SEE COVER PAGE'' message
CR or 5	Copyright notice
UW or 6	Unpublished Work

An unrecognized *type* will default to the PROPRIETARY marking.

By default, the copyright notice (CR) and the ''Unpublished Work'' notice (UW) use the current year as the copyright date. A year may be supplied as the second argument to .PM to override the copyright date.

See Appendix G for a sample of each type of marking.

9.8 Private Documents

`.nr Pv` *value*

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

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

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

10. 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.

These macros should normally appear only once at the end of the document, after the signature block (`.SG`) and notations (`.NS`/`.NE`) macros. They 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 is often not needed, and is therefore produced at the end.

10.1 Table of Contents

`.TC` [*slevel*] [*spacing*] [*tlevel*] [*tab*] [*head1*] [*head2*] [*head3*] [*head4*] [*head5*]

The `.TC` macro generates a table of contents containing the headings that were saved for the table of contents as determined by the value of the `Cl` register. The arguments to `.TC` control the spacing before each entry, the placement of the associated page number, and additional text on the first page of the table of contents before the word ''CONTENTS'' is printed.

Spacing before each entry is controlled by the first two arguments; headings whose level is less than or equal to *slevel* will have *spacing* blank vertical spaces before them. Both *slevel* and *spacing* default to 1. This means that first-level headings are preceded by one blank vertical space. Note that *slevel* does not control what levels of heading have been saved; the saving of headings is the function of the `Cl` register.

The third and fourth arguments control the placement of the page number of each heading. The page numbers can be justified at the right margin with either blanks or dots 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 is 2), the page numbers are justified at the right margin. In this case, the value of *tab* determines the character used to separate the heading text from the page number. If *tab* is 0 (the default value), dots (i.e., leaders) are used; if *tab* is greater than 0, spaces are used. For headings whose level is greater than *tlevel*, the page numbers are separated from the heading text by two spaces (i.e., they are ''ragged right'').

All additional arguments (e.g., *head1*, *head2*), if any, are horizontally centered on the page, and precede the actual table of contents itself.

If the `.TC` macro is invoked with at most four arguments, then the user-exit macro `.TX` is invoked (without arguments) before the word ''CONTENTS'' is printed, or the user-exit macro `.TY` is invoked 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 +10n
Approved:  \l'3i'
.in
.sp
..
.TC
```

yields:

Special Application
Message Transmission

Approved: ____

CONTENTS

If this macro were defined as `.TY` rather than `.TX`, the word ‘‘CONTENTS’’ would not appear. 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 at the left margin. Subsequent levels will be aligned with the text of headings at the preceding level. These indentations 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. The arguments must be scaled. For example, if `Cl` is 5, `Ci` would be defined as:

```
.ds Ci .25i .5i .75i 1i 1i
```

or

```
.ds Ci 0 2n 4n 6n 8n
```

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 lower-case 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 place the page number. Ordinarily, the same `.PF` used in the body of the document will be adequate.

The List of Figures, List of Tables, List of Exhibits, and List of Equations 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.

10.2 Cover Sheet

For document styles that allow a cover sheet (e.g., released-paper and memoranda), the format of the cover sheet resembles the first-page format of the document and includes the abstract. Cover sheets are not generated for external letters and business letters.

11. REFERENCES

There are two macros that delimit the text of references, a string used to automatically number the references, and an optional macro to produce the reference page(s) within the document. The input format for a reference is:

> *segment of text to be referenced*`\*(Rf`
> `.RS` [*string-name*]
> *one or more lines of*
> *reference text*
> `.RF`

The string `Rf` generates the reference mark. The `.RS` and `.RF` macros delimit the text of each reference. The reference text is saved and printed on the reference page at the end of the document. The argument *string-name* may be used to keep track of a particular reference mark for subsequent references.

11.1 Automatic Numbering of References

The string call `\*(Rf` generates an automatically-numbered reference mark in the text. The reference mark consists of the reference number enclosed in square brackets, and is printed in a smaller size and raised (e.g., superscripted).

The register `:R` keeps track of the current reference number. The string `Rf` automatically increments `:R` and then uses that value as the reference number in the reference mark. The `:R` register may have its format or value changed to affect the reference marks.

11.2 Subsequent References

The `.RS` macro accepts one argument (*string-name*) and uses it as the name of string for subsequent references. If given, *string-name* is assigned the current reference mark and can be used later in the document to reference text which must be labeled with a prior reference number. No `.RS`/`.RF` pair is needed for subsequent references marked in the text. For example, if the input for a reference was:

```
and the ABC package\*(Rf
.RS aA
A. B. Smith and J. R. Doe,
.ul
The ABC Package \(em Reference Manual,
SoftPackagers, Inc.
.RF
```

and that reference was numbered ‘‘4’’, then later in the document, the input:

```
is available with ABC\*(aA as well.
```

produces:

> is available with ABC[4] as well.

11.3 Reference Page

A reference page will be generated automatically at the end of the document (before the table of contents and the cover sheet, if any) and will be listed in the table of contents. This page contains the reference items (i.e., text enclosed in `.RS`/`.RF` pairs). Reference items will be separated by one blank vertical space; if the register `Ls` is set to 0, this spacing is suppressed.

The string `Rp` contains the reference page title. By default, ‘‘REFERENCES’’ is used as the title. The user may change the reference page title by defining the `Rp` string:

```
.ds Rp List of References
```

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

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

The two 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 `.RP` does not issue a `.SK`, then a single blank vertical space will separate the references from the preceding or following text. The user may wish to adjust the spacing. For example:

```
.sp 3
.RP 1 2
.H 1 "Next Section"
```

produces references at the end of each major section.

12. MISCELLANEOUS FEATURES

12.1 Using Fonts for Emphasis

`.B` [*bold*] [*previous*] [...]

`.I` [*bold*] [*previous*] [...]

`.R`

The MM package recognizes three standard fonts or typefaces: roman, italic, and bold. The roman font is the typeface used for regular text. When called without arguments, the `.B` and `.I` macros change to the bold and italic fonts, respectively. The font change remains in effect until another call to `.B` or `.I`, or until the roman (regular text) font is restored with the `.R` macro. For example:

```
This is the roman font.
.B
Here is some text in the bold font,
.I
and here is some text in the italic font.
.R
Now back to the regular font.
```

yields:

This is the roman font. **Here is some text in the bold font,** *and here is some text in the italic font.* Now back to the regular font.

If `.B` is called with one argument, that argument is printed in the bold font; likewise, if `.I` is called with one argument, that argument is printed in the italic font. Then the previous font is restored. If two or more arguments (maximum 6) are given to a `.B` or a `.I` macro, the second argument is then concatenated to the first with no intervening space (a narrow space is inserted after italics), but is printed in the previous font; the remaining pairs of arguments are similarly alternated. For example:

```
.I abc def ghi jkl
```

produces:

*abc*def*ghi*jkl

Use the following macros to alternate between specific pairs of fonts:

`.IB` [*italic*] [*bold*] [...]

`.BI` [*bold*] [*italic*] [...]

`.IR` [*italic*] [*roman*] [...]

`.RI` [*roman*] [*italic*] [...]

`.RB` [*roman*] [*bold*] [...]

`.BR` [*bold*] [*roman*] [...]

Each takes a maximum of 6 arguments and alternates the arguments between the specified fonts. Note the font changes in headings are handled separately.

12.2 Right Margin Justification

`.SA` [*mode*]

The `.SA` macro is used to set right-margin justification for the main body of text. Two justification flags are used: ‘‘current’’ and ‘‘default.’’ `.SA 0` sets both flags to no justification, i.e., it acts like the `.na` request and produces a ragged right margin. `.SA 1` is the inverse: it sets both flags to cause justification, just like the `.ad` request. However, calling `.SA` without an argument causes the ‘‘current’’ flag to be copied from the ‘‘default’’ flag, thus performing either `.na` or `.ad`, depending on what the default is. Initially, both flags are set for justification.

In general, the `.na` request can be used to ensure that justification is turned off, but `.SA` should be used to restore justification, rather than the `.ad` request. In this way, justification or lack thereof for the remainder of the text is specified by inserting `.SA 0` or `.SA 1` once at the beginning of the document.

12.3 Two-Column Output

MM can print two columns of text on a page:

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

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

12.3.1 Width Control for Footnotes and Displays

`.WC` [*code*]

It is possible to have full-page width footnotes and displays when in two-column mode, although the default action is for footnotes and displays to be narrow in two-column mode and wide in one-column mode. The width of footnotes and displays are controlled by the `.WC` (width control) macro which takes the following arguments:

code	Meaning
`N`	Use normal default mode (`-WF`, `-FF`, `-WD`, `FB`).
`WF`	Always use wide footnotes, even in two-column mode.
`-WF`	Default: Turn off `WF`. Footnotes follow the column mode in effect (wide in one-column mode and narrow in two-column mode) unless `FF` is set.
`FF`	All footnotes have the same width as the first footnote for that page.
`-FF`	Default: Turn off `FF`. The footnote style follows the settings of `WF` or `-WF`.
`WD`	Always use wide displays, even in two-column mode.
`-WD`	Default: Turn off `WD`. Displays follow the column mode in effect when the 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.

For example:

```
.WC WD FF
```

will cause all displays to be wide, and all footnotes on a page to be the same width, while:

```
.WC N
```

will reinstate the default actions. If conflicting settings are given to `.WC`, the last one is used. That is, ``.WC WF -WF'' has the effect of ``.WC -WF''.

12.3.2 Column Headings for Two-Column Output •

In two-column 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 to provide header lines both for the entire page and for each of the columns. For example:

```
.de TP
.sp 2
.tl 'Page \\nP'OVERALL''
.tl ''TITLE''
.sp
.nf
.ta 1.5iC 3iR 3.5i 5iC 6.5iR
left-part➤center-part➤right-part➤left-part➤center-part➤right-part
left-part➤center-part➤right-part➤left-part➤center-part➤right-part
.fi
.sp2
..
```

(where ➤ represents the tab character). The above example will produce two lines of page header text plus two lines of headers over each column. The tab stops are for a 6.5-inch overall line length.

12.4 Blank Vertical Spaces

`.SP` [*lines*]

There exist several ways of obtaining blank vertical spaces, all with different effects. The `.sp` request spaces the number of lines specified, unless ``no space'' (`.ns`) mode is on, in which case the request is ignored. This mode is set at the end of a page header to eliminate spacing by a `.sp` or a `.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 blank vertical space by successive macro calls. Several `.SP` calls in a row produce not the sum of their arguments, but their maximum. For example:

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

produces only 3 blank lines. Many MM macros utilize `.SP` for spacing. For example, `.LE 1` immediately followed by `.P` produces only a single blank vertical space between the end of the list and the following paragraph.

An omitted argument defaults to one blank vertical space. Negative arguments are not permitted. The argument must be unscaled but fractional amounts are permitted. Like `.sp`, `.SP` is also inhibited by the `.ns` request.

12.5 Skipping Pages

`.SK` [*pages*]

The `.SK` macro skips pages, but retains the usual header and footer processing. If the argument *pages* is omitted, null, or 0, `.SK` skips to the top of the next page unless it is currently at the top of a page, in which case it does nothing. If the argument *pages* is given, `.SK` *n* skips *n* pages, that is, `.SK` always positions the text that follows it at the top of a page, while `.SK 1` always leaves one page that is blank except for the header and footer.

12.6 Forcing an Odd Page

`.OP`

This macro is used to ensure that the text following it begins at the top of an odd-numbered page. If currently at the top of an odd page, no action 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.

12.7 Setting the Point Size and Vertical Spacing

The default point size (obtained from the MM register `S`) is 10 points, and the vertical spacing is 12 points. The prevailing point size and vertical spacing may be changed by invoking the `.S` macro:

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

The mnemonics `D` (default value), `C` (current value), and `P` (previous value) may be used for both arguments. If an argument is negative, the current value is decremented by the specified amount; if an argument is positive, the current value is incremented by the specified amount; if an argument is unsigned, it is used as the new value; `.S` without arguments defaults to `P`. If the first argument is specified but the second is not, then `D` is used for the vertical spacing; the default value for vertical spacing is always 2 points greater than the current point size. A null (`""`) value for either argument defaults to `C`. Thus, if *n* is a numeric value:

`.S`	is equivalent to	`.S P P`
`.S ""` *n*	is equivalent to	`.S C` *n*
`.S` *n* `""`	is equivalent to	`.S` *n* `C`
`.S` *n*	is equivalent to	`.S` *n* `D`
`.S ""`	is equivalent to	`.S C D`
`.S "" ""`	is equivalent to	`.S C C`
`.S` *n n*	is equivalent to	`.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 2 points) is used. For example:

`.S 100`	is equivalent to	`.S 10 12`
`.S 14 111`	is equivalent to	`.S 14 16`

12.8 Size Reduction

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

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

```
.SM X            gives   X
.SM X Y          gives   XY
.SM Y X Y        gives   YXY
.SM UNIX         gives   UNIX
.SM UNIX )       gives   UNIX)
.SM ( UNIX )     gives   (UNIX)
.SM U NIX ""     gives   UNIX
```

12.9 Producing Accents

Seven strings are defined to place accent marks with letters. The accent string immediately follows the letter that it is used with:

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

12.10 Inserting Text Interactively •

`.RD` [*prompt*] [*diversion-name*] [*string-name*]

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 newlines are found. When the newlines are encountered, normal output is resumed. The `.RD` macro follows the formatting conventions in effect. Thus, the examples below assume that `.RD` is invoked in no-fill mode.

The first argument is a prompt which will be printed at the terminal. If *prompt* is not given, `.RD` signals the user with a BEL (bell sound). The second argument, the name of a diversion, allows the user to save all the text typed in after the prompt in a macro named *diversion-name*. The third argument, the name of a string, allows the user to save the first line following the prompt in the string named *string-name* for later reference. For example:

```
.RD NAME aA bB
```

produces the prompt ``NAME'' after which the user types in the information to be read and ends with an extra line return:

```
NAME  J. Jones
16 Elm Road
Piscataway, N.J.
newline
```

The diverted macro `aA` will contain:

J. Jones
16 Elm Road
Piscataway, N.J.

The string `bB` contains ‘‘J. Jones’’. The collected information can then be used by invoking `.aA` for the diverted macro and `\*(bB` for the defined string.

A newline followed by a control-d (EOF) also allows the user to resume normal output.

13. ERRORS AND DEBUGGING

13.1 Error Terminations

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

- A break occurs.
- To avoid confusion regarding the location of the error, the formatter output buffer (which may contain some text) is printed.
- A brief message is printed giving the name of the macro that found the error, the type of error, and the approximate line number in the current input file of the last processed input line. (All of the error messages are explained in Appendix D.)
- Processing terminates, unless register `D` has a positive value. In the latter case, processing continues even though the output is guaranteed to be deranged from that point on.

A harmless ‘‘broken pipe’’ message may result if any of the preprocessors (*eqn*, *tbl*, *pic*, *grap*) are used and if the *troff* option –`o`*list* causes the last page of the document not to be printed.

The following error conditions with the business letter macros (see §6.3) will cause processing to abort:

- if you omit the `.LT` macro,
- if you use an unrecognized code as an argument to the `.LT` macro,
- if you omit either or both of the `.WA` and `.WE` macros,
- if you omit either or both of the `.IA` and `.IE` macros,
- if you use the `.LO` macro without an argument or if the argument is an unrecognized code.

In each case, an appropriate error message will be printed.

13.2 Disappearance of Output

This usually occurs because of an unclosed diversion (e.g., missing `.DE` or `.FE`). Fortunately, the macros that use diversions are careful about it, and they check to make sure that illegal nesting does not occur. If a message is issued about a missing `.DE` or `.FE`, the appropriate action is to search backwards from the termination point looking for the corresponding `.DF`, `.DS` or `.FS`.

Use the *checkdoc* command to check for illegal nesting and/or omission of these macros.

13.3 The *checkdoc* Command

For a quick check on proper usage of the MM macros, use the command:

```
checkdoc filename
```

The *checkdoc* command finds mismatched macro pairs, missing arguments, macros out of sequence, or any other misuse of the MM macros as well as mismatched delimiters for *eqn* and other miscellaneous error conditions that cause problems in formatting. If a problem is encountered, *checkdoc* prints an appropriate message with the input filename and the line number where the problem occurs.

14. EXTENDING AND MODIFYING THE MACROS •

14.1 Naming Conventions

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

n	digit
a	lower-case letter
A	upper-case letter
x	*n*, *a* or *A* : i.e., any letter or digit (any alphanumeric character)
s	special character (any non-alphanumeric character)

All other characters are literals (i.e., stand for themselves).

The formatter keeps the names of requests, macros, and strings in a single internal table, so there must be no duplication among such names. Names for number registers are kept in a separate table.

14.1.1 Names Used by Formatters

Requests:

aa	most common
an	only one, currently `c2`

Registers:

aa	normal
`.`*a*	normal
`.`*A*	normal
`.`*s*	only one, currently `.$`
a`.`	only one, currently `c.`
`%`	page number

14.1.2 Names Used by MM

Macros and Strings:

A AA Aa	accessible to users (e.g., macros `.P` and `.HU`; strings `F` and `Rf`)
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 in §12.9
`)`*x* `}`*x* `]`*x* `>`*x* `?`*x*	internal

Registers:

An Aa	accessible to users (e.g., `H1` and `Fg`)
A	accessible to users; set on the command line (e.g., `W`)
`:`*x* `;`*x* `#`*x* `?`*x* `!`*x*	internal

14.1.3 Names Used by *eqn* and *tbl*. The equation preprocessor, *eqn*, uses 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* `^`*a* `#`*a* `#`*s*

14.1.4 User-Definable Names. Given the above, what is left for user extensions? To avoid collisions with existing names, use names that consist either of a single lower-case letter, or of a lower-case letter followed by a character other than a lower-case letter (remembering, however, that the names `.c2` and `.nP` are already used). The following is a possible user naming convention:

Macros:	*aA*	(e.g., `bG`, `kW`)
Strings:	*as*	(e.g., `c)`, `f]`, `p}`)
Registers:	*a*	(e.g., `f`, `t`)

14.2 Sample Extensions

14.2.1 Appendix Headings. The following is a way of generating and numbering appendix headings:

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

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

`.aH` *title*

begins a new page with the header changed to ‘‘Appendix *a-P*’’ where *a* is the appendix number and *P* is the page number. The appendix title is generated as an unnumbered, centered heading which can be saved for the table of contents, if desired.

14.2.2 Hanging Indent with Tabs. The following example illustrates the use of the hanging indent feature of variable-list items. First, a user-defined macro is built 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 ‘‘`\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 macro and an example of its use are:

```
.de aX
.LI
\&\\$1\t\\$2\t\\$3\t\\$4\t\c
..
 ...
.ta 9n 18n 27n 36n
.VL 36
.aX .nh off \- no
No hyphenation.
Automatic hyphenation is turned off.
Words containing hyphens (e.g., mother-in-law)
may still be split across lines.
.aX .ht on \- no
Hyphenate.
Automatic hyphenation is turned on.
.aX .hc\ c none none no
Hyphenation indicator character is set to ''c'' or removed.
During text processing the indicator is suppressed
and will not appear in the output.
Prepend the indicator to a word has the effect
of preventing hyphenation of that word.
.LE
```

* The *troff* escape sequence `\&` produces a ‘‘zero-width’’ space, i.e., it causes no output characters to appear, but it removes the special meaning of a leading period or apostrophe.

The resulting output is:

.nh	off	–	no	No hyphenation. Automatic hyphenation is turned off. Words containing hyphens (e.g., mother-in-law) may still be split across lines.
.ht	on	–	no	Hyphenate. Automatic hyphenation is turned on.
.hc c	none	none	no	Hyphenation indicator character is set to ‘‘c’’ or removed. During text processing the indicator is suppressed and will not appear in the output. Prepend the indicator to a word has the effect of preventing hyphenation of that word.

APPENDIX A: Summary of Differences for MM with NROFF

Some MM features behave differently with the *nroff* text formatter than with *troff*. The *nroff* behavior is noted below along with the section number in the manual where the feature is discussed.

General

- All values are unscaled and are treated as the number of character positions (e.g., spaces) or vertical lines, as appropriate.
- Each reference to ‘‘blank vertical space’’ means a full vertical line. [§1.2]
- By default, tabs are set every 8 character spaces. [§3.5]
- Not all formatter requests are available with *nroff*. For example, the `.fp`, `.lg`, and `.ss` requests are for use with *troff* only. [§3.8]
- Text in the italic font is underlined; text in the bold font is emboldened by overstriking. [§4.2.2.4, §12.1]
- Point size and vertical spacing are for use with *troff* only and are ignored by *nroff*. [§4.2.2.5, §12.4, §12.7]
- Right-margin justification is turned off. By default, the ‘‘current’’ and ‘‘default’’ justification flags are set to no justification; initially, justification is set with `.SA 0`. [§12.2]

Typical Command Lines [§2.2]

- Plain text (no tables or equations):

```
nroff -mm [options] filename | ...
```

- Text with tables:

```
tbl filename | nroff -mm [options] | ...
```

- Text with equations:

```
neqn /usr/pub/eqnchar filename | nroff -mm [options] | ...
```

- Text with both tables and equations:

```
tbl filename | neqn /usr/pub/eqnchar - | nroff -mm [options] | ...
```

Preprocessing by *tbl* and *neqn*, if needed, must be invoked as shown in the command line prototypes. If used, *neqn* immediately precedes *nroff*. (The *grap* and *pic* preprocessors are for use with *troff* only.)

When formatting a document with *nroff*, the output should normally be processed for a specific type of terminal, because the output may require some features that are specific to a given terminal, e.g., reverse paper motion or half-line paper motion in both directions. Output for any terminal incapable of reverse paper motion and also lacking hardware tab stops should be processed as follows:

```
nroff -mm -T745 filename | col -x
```

If two-column processing is used with *nroff*, the *nroff* output must be postprocessed by *col*. In the latter case, the `–T37` terminal type must be specified to *nroff* and the output of *col* must be processed by the appropriate terminal filter (e.g., *450*). For example, two-column output for a DIABLO 450S terminal should be processed as follows:

```
nroff -mm –T37 filename | col -x | 450
```

To produce boxed tables on printers without fractional line capabilities, use *tbl* with the `-TX` option.

Parameters that Can Be Set from the Command Line [§2.3]

`-rA2` suppresses the letterhead block to accommodate preprinted stationary.

`-rT`*n* provides register settings for certain devices. If *n* is 1, the page width (e.g., line length) and page offset are set to 80 and 3 character spaces, respectively. If *n* is 2, the page length is set to 84 lines and underlining is turned off. By default, *n* is 0 for standard page length, page width and page offset.

`-rU1` controls underlining of section headings. This flag causes only letters and digits to be underlined; otherwise all characters (including spaces) are underlined.

By default, the page length is set to 66 lines (e.g., `-rL66`), the page offset is set to 10 character spaces (e.g., `-rO10`), and the page width is set to 60 character spaces (e.g., `-rW60`). The page width for 12-pitch type would be 72 character spaces (e.g., `-rW72`).

Strings for Special Symbols

- The string `\*(BU` produces a real bullet (•) on printers that have one. Otherwise, a bullet is represented as ⊕ (by overstriking ‘‘o’’ and ‘‘+’’). [§3.7.1]
- Most *nroff* printers do not offer distinct graphics for a dash, a minus sign, and a hyphen. The string `\*(EM` represents an em dash as two hyphens (--). A hyphen is just the terminal dash (-) character. The sequence `\-` produces a real minus sign (–) on printers that have one; otherwise a hyphen is used. [§3.7.2]
- The strings `\*(Rg`, `\*(Tm`, and `\*(Sm` produce the registered mark, trademark, and service mark symbols, respectively, on printers that have them. Otherwise, the registered mark, trademark, and service mark symbols are represented as (R), (TM), and (SM), respectively; in this case, the mark is raised one-half line for printers that have half-line capabilities. [§3.7.3]

Paragraphs and Headings

- By default, paragraph indentation is 5 character spaces; initially, register `Pi` is set to 5. [§4.1]
- By default, all heading levels are underlined; string `HF` is set to all 2’s. [§4.2.2.4]

Lists

- By default, list item indentation for an automatically-numbered list is 6 character spaces; initially, register `Li` is set to 6. [§5.3.3.1]
- By default, list item indentation for a reference list is 6 character spaces. [§5.3.3.5]

Displays [§7.1, §7.2]

- By default, display indentation (for display *format* `2` or `I`) is 5 character spaces; initially, register `Si` is set to 5.
- The *rindent* argument must be an unscaled number and is treated as character spaces.

Footnotes

By default, footnote text is formatted with no hyphenation and no justification; initially, the footnote style is set to `.FD 10`. [§8.3]

References

The reference mark appears full-sized on the main line and is separated from the preceding text by one space (e.g., segment of text to be referenced [1] continue). [§11.1]

APPENDIX B: User-Defined List Structures •

If a large document requires complex list structures, it is useful to be able to define the appearance for each list level only once, instead of having to define it at the beginning of each list. This permits consistency of style in a large document. For example, a generalized list-initialization macro might be defined in such a way that what it does depends on the list-nesting level in effect at the time the macro is called. Suppose that levels 1 through 5 of lists are to 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, you should know that the number register `:g` is used by the MM list macros to determine the current list level, and 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.

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

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
```

will yield:

```
A.  first line.

       [1]  second line.

B.  third line.
```

There is another approach to lists that is similar to the `.H` mechanism. The list-initialization, as well as the `.LI` and `.LE` macros, are all included in a single macro. That macro (called `.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
'        \" if there is an argument, that is the level
.ie \\n(.$ .nr g \\$1
'        \" if no argument, use current level
.el .nr g \\n(:g
'        \" increasing level by more than 1 is an error
.if \\ng-\\n(:g>1 .)D "*** ILLEGAL SKIPPING OF LEVEL"
'        \" if g is greater than :g, begin new list
'        \" and reset g to current level (.aL changes g)
.if \\ng>\\n(:g \{.AL \\ng-1
.nr g \\n(:g \}
'        \" if :g is greater than g, prune back to correct level
.if \\n(:g>\\ng .LC \\ng
'        \" if :g equals g, stay within current list
'        \" in all cases, get out an item
.LI
..
```

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 for `:g`. Invoking `.bL` without arguments causes it to stay at the current list level. The MM `.LC` macro (list clear) 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`. 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 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.
```

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.

APPENDIX C: Sample Memorandum, Released Paper, and External Letter

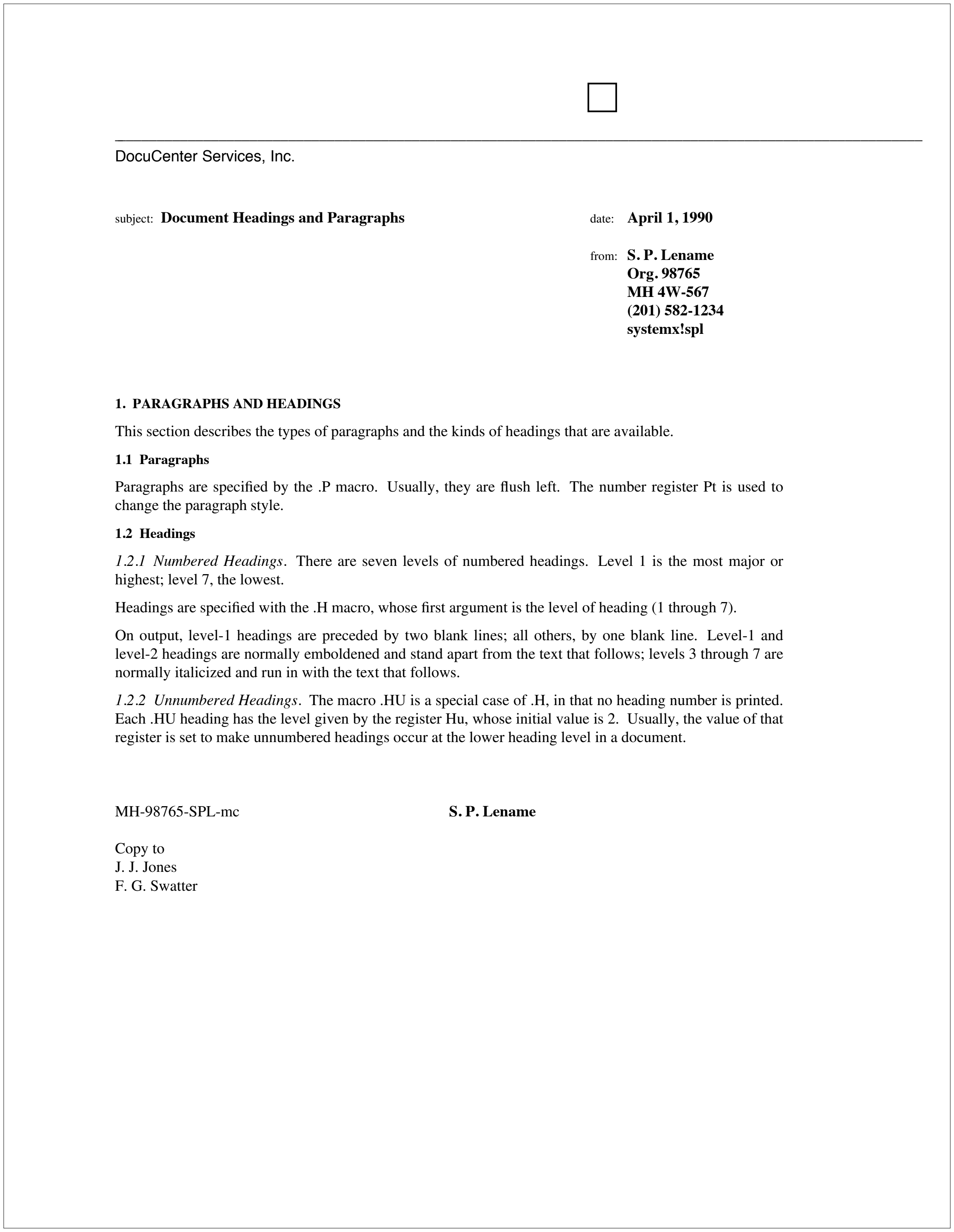
This appendix demonstrates the use of MM in generating various document styles. The information needed for memoranda, released papers, and external letters is entered in the same way. In this appendix, the input for all three documents varies in only one respect — a different `.MT` type is used to select the document style:

`.MT 0`	Memorandum
`.MT 4`	Released Paper
`.MT 5`	External Letter

The following input was used to produce the sample documents shown on the next three pages.

```
.ND "April 1, 1990"
.TL
Document Headings and Paragraphs
.AF "DocuCenter Services, Inc."
.AU "S. P. Lename" SPL MH 98765 "(201) 582-1234" 4W-567 systemx!spl
.MT type
.H 1 "PARAGRAPHS AND HEADINGS"
This section describes the types of paragraphs and the
kinds of headings that are available.
.H 2 Paragraphs
Paragraphs are specified by the .P macro.
Usually, they are flush left.
The number register Pt is used to change the paragraph style.
.H 2 Headings
.H 3 "Numbered Headings."
There are seven levels of numbered headings.
Level 1 is the most major or highest;
level 7, the lowest.
.P
Headings are specified with the .H macro,
whose first argument is the level of heading
(1 through 7).
.P
On output, level-1 headings are preceded by two blank lines;
all others, by one blank line.
Level-1 and level-2 headings are normally emboldened and
stand apart from the text that follows;
levels 3 through 7 are normally italicized and run in with the
text that follows.
.H 3 "Unnumbered Headings."
The macro .HU is a special case of .H, in that no heading
number is printed.
Each .HU heading has the level given by the register Hu,
whose initial value is 2.
Usually, the value of that register is set to make
unnumbered headings occur at the lower heading level
in a document.
.SG mc
.NS
J. J. Jones
F. G. Swatter
.NE
```

OUTPUT: Memorandum (`.MT 0`)



DocuCenter Services, Inc.

subject: **Document Headings and Paragraphs**

date: **April 1, 1990**

from: **S. P. Lename**
Org. 98765
MH 4W-567
(201) 582-1234
systemx!spl

1. PARAGRAPHS AND HEADINGS

This section describes the types of paragraphs and the kinds of headings that are available.

1.1 Paragraphs

Paragraphs are specified by the .P macro. Usually, they are flush left. The number register Pt is used to change the paragraph style.

1.2 Headings

1.2.1 Numbered Headings. There are seven levels of numbered headings. Level 1 is the most major or highest; level 7, the lowest.

Headings are specified with the .H macro, whose first argument is the level of heading (1 through 7).

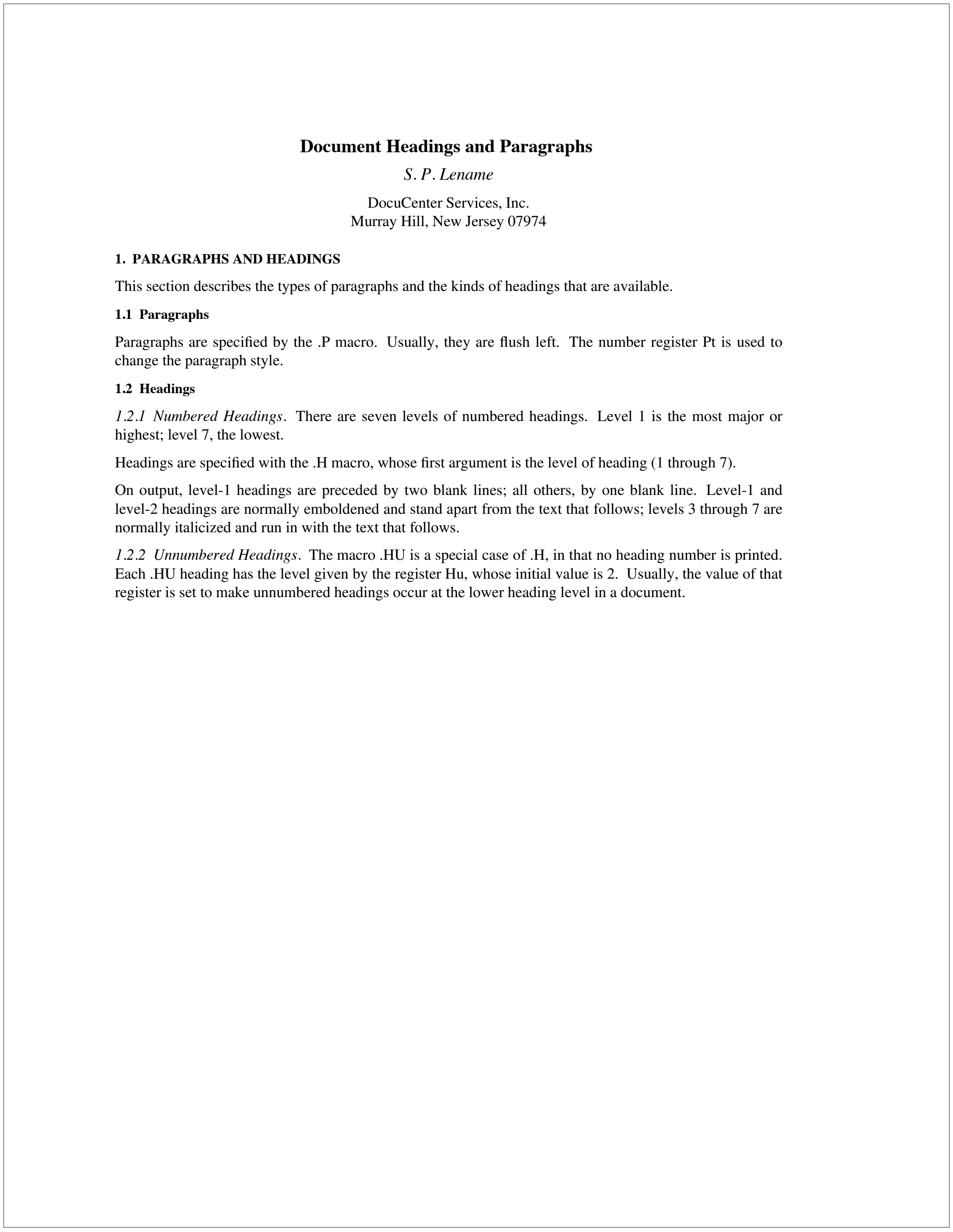
On output, level-1 headings are preceded by two blank lines; all others, by one blank line. Level-1 and level-2 headings are normally emboldened and stand apart from the text that follows; levels 3 through 7 are normally italicized and run in with the text that follows.

1.2.2 Unnumbered Headings. The macro .HU is a special case of .H, in that no heading number is printed. Each .HU heading has the level given by the register Hu, whose initial value is 2. Usually, the value of that register is set to make unnumbered headings occur at the lower heading level in a document.

MH-98765-SPL-mc **S. P. Lename**

Copy to
J. J. Jones
F. G. Swatter

OUTPUT: Released Paper (`.MT 4`)



Document Headings and Paragraphs

S. P. Lename

DocuCenter Services, Inc.
Murray Hill, New Jersey 07974

1. PARAGRAPHS AND HEADINGS

This section describes the types of paragraphs and the kinds of headings that are available.

1.1 Paragraphs

Paragraphs are specified by the .P macro. Usually, they are flush left. The number register Pt is used to change the paragraph style.

1.2 Headings

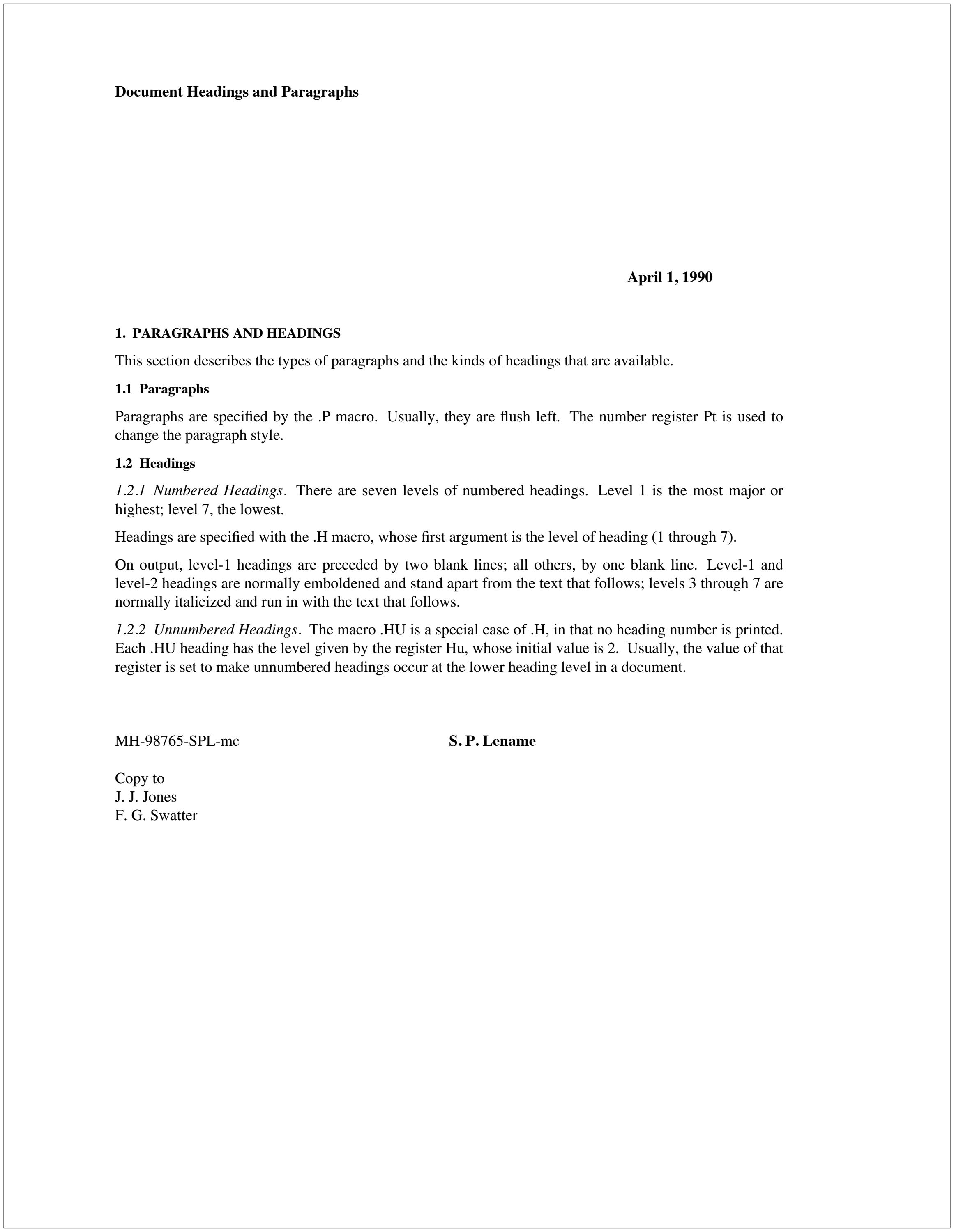
1.2.1 Numbered Headings. There are seven levels of numbered headings. Level 1 is the most major or highest; level 7, the lowest.

Headings are specified with the .H macro, whose first argument is the level of heading (1 through 7).

On output, level-1 headings are preceded by two blank lines; all others, by one blank line. Level-1 and level-2 headings are normally emboldened and stand apart from the text that follows; levels 3 through 7 are normally italicized and run in with the text that follows.

1.2.2 Unnumbered Headings. The macro .HU is a special case of .H, in that no heading number is printed. Each .HU heading has the level given by the register Hu, whose initial value is 2. Usually, the value of that register is set to make unnumbered headings occur at the lower heading level in a document.

OUTPUT: External Letter (`.MT 5`)



Document Headings and Paragraphs

April 1, 1990

1. PARAGRAPHS AND HEADINGS

This section describes the types of paragraphs and the kinds of headings that are available.

1.1 Paragraphs

Paragraphs are specified by the .P macro. Usually, they are flush left. The number register Pt is used to change the paragraph style.

1.2 Headings

1.2.1 Numbered Headings. There are seven levels of numbered headings. Level 1 is the most major or highest; level 7, the lowest.

Headings are specified with the .H macro, whose first argument is the level of heading (1 through 7).

On output, level-1 headings are preceded by two blank lines; all others, by one blank line. Level-1 and level-2 headings are normally emboldened and stand apart from the text that follows; levels 3 through 7 are normally italicized and run in with the text that follows.

1.2.2 Unnumbered Headings. The macro .HU is a special case of .H, in that no heading number is printed. Each .HU heading has the level given by the register Hu, whose initial value is 2. Usually, the value of that register is set to make unnumbered headings occur at the lower heading level in a document.

MH-98765-SPL-mc **S. P. Lename**

Copy to
J. J. Jones
F. G. Swatter

APPENDIX D: Sample Business Letters

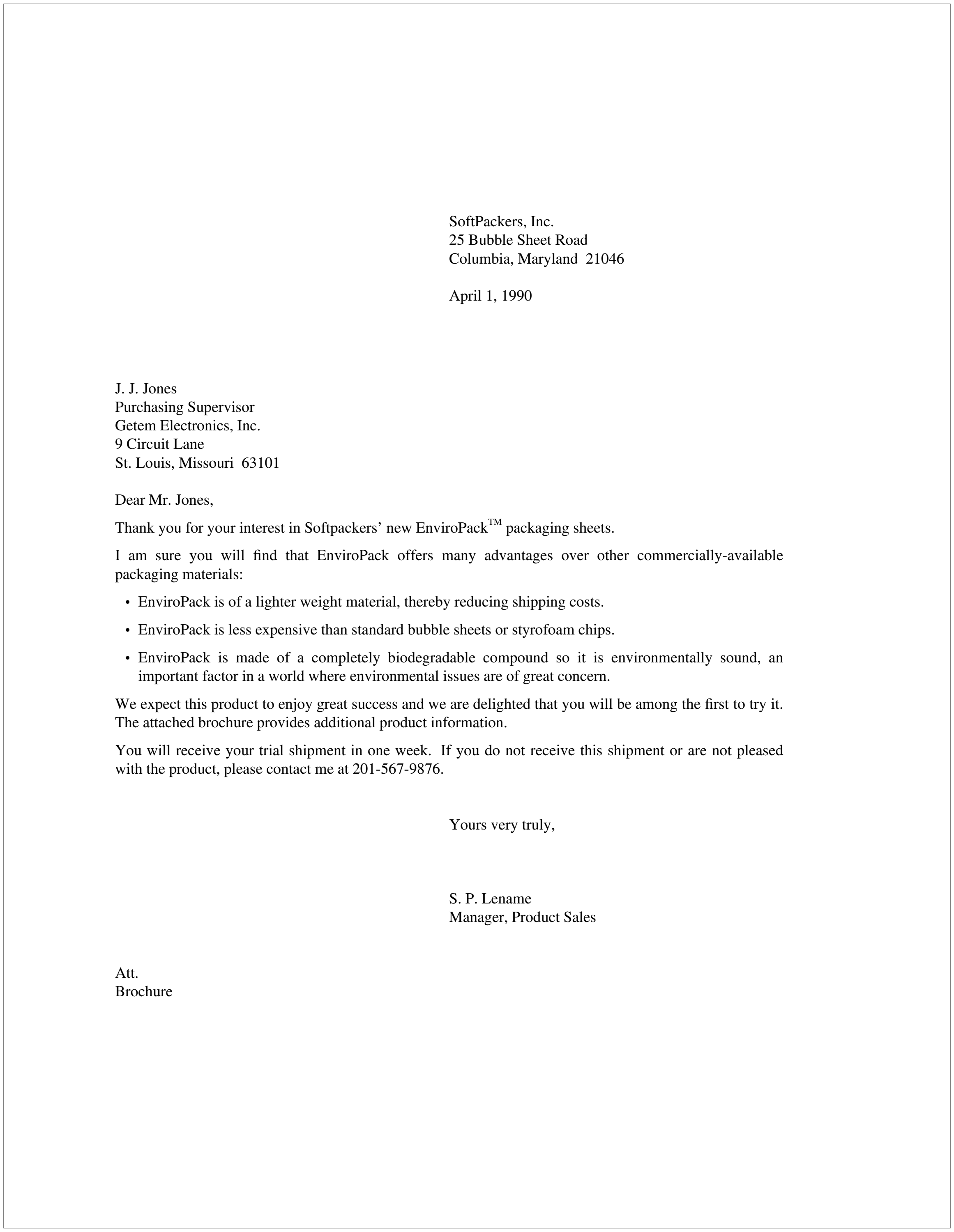
This appendix demonstrates the use of MM in generating common business letters. The information needed for the four formats (blocked, semi-blocked, full-blocked, simplified) is entered in the same way. In this appendix, the input for all four documents varies in only one respect — a different `.LT` type is used to select the business letter format:

`.LT BL`	Blocked Letter
`.LT SB`	Semi-Blocked Letter
`.LT FB`	Full-Blocked Letter
`.LT SP`	Simplified Letter

The following input was used to produce the sample business letters shown on the next four pages.

```
.ND "April 1, 1990"
.WA "S. P. Lename" "Manager, Product Sales"
SoftPackers, Inc.
25 Bubble Sheet Road
Columbia, Maryland  21046
.WE
.IA "J. J. Jones" "Purchasing Supervisor"
Getem Electronics, Inc.
9 Circuit Lane
St. Louis, Missouri  63101
.IE
.LO SA "Dear Mr. Jones,"
.LT type
.P
Thank you for your interest in Softpackers’ new
EnviroPack\*(Tm packaging sheets.
.P
I am sure you will find that EnviroPack offers many advantages
over other commercially-available packaging materials:
.BL
.LI
EnviroPack is of a lighter weight material, thereby reducing
shipping costs.
.LI
EnviroPack is less expensive than standard bubble sheets or
styrofoam chips.
.LI
EnviroPack is made of a completely biodegradable
compound so it is environmentally sound, an important factor
in a world where environmental issues are of great concern.
.LE
.P
We expect this product to enjoy great success and we are
delighted that you will be among the first to try it.
The attached brochure provides additional product information.
.P
You will receive your trial shipment in one week.
If you do not receive this shipment or are not pleased
with the product, please contact me at 201-567-9876.
.FC
.SG
.NS 3
Brochure
.NE
```

OUTPUT: Blocked Business Letter (`.LT BL`)



SoftPackers, Inc.
25 Bubble Sheet Road
Columbia, Maryland 21046

April 1, 1990

J. J. Jones
Purchasing Supervisor
Getem Electronics, Inc.
9 Circuit Lane
St. Louis, Missouri 63101

Dear Mr. Jones,

Thank you for your interest in Softpackers' new EnviroPack$^{\text{TM}}$ packaging sheets.

I am sure you will find that EnviroPack offers many advantages over other commercially-available packaging materials:

- EnviroPack is of a lighter weight material, thereby reducing shipping costs.
- EnviroPack is less expensive than standard bubble sheets or styrofoam chips.
- EnviroPack is made of a completely biodegradable compound so it is environmentally sound, an important factor in a world where environmental issues are of great concern.

We expect this product to enjoy great success and we are delighted that you will be among the first to try it. The attached brochure provides additional product information.

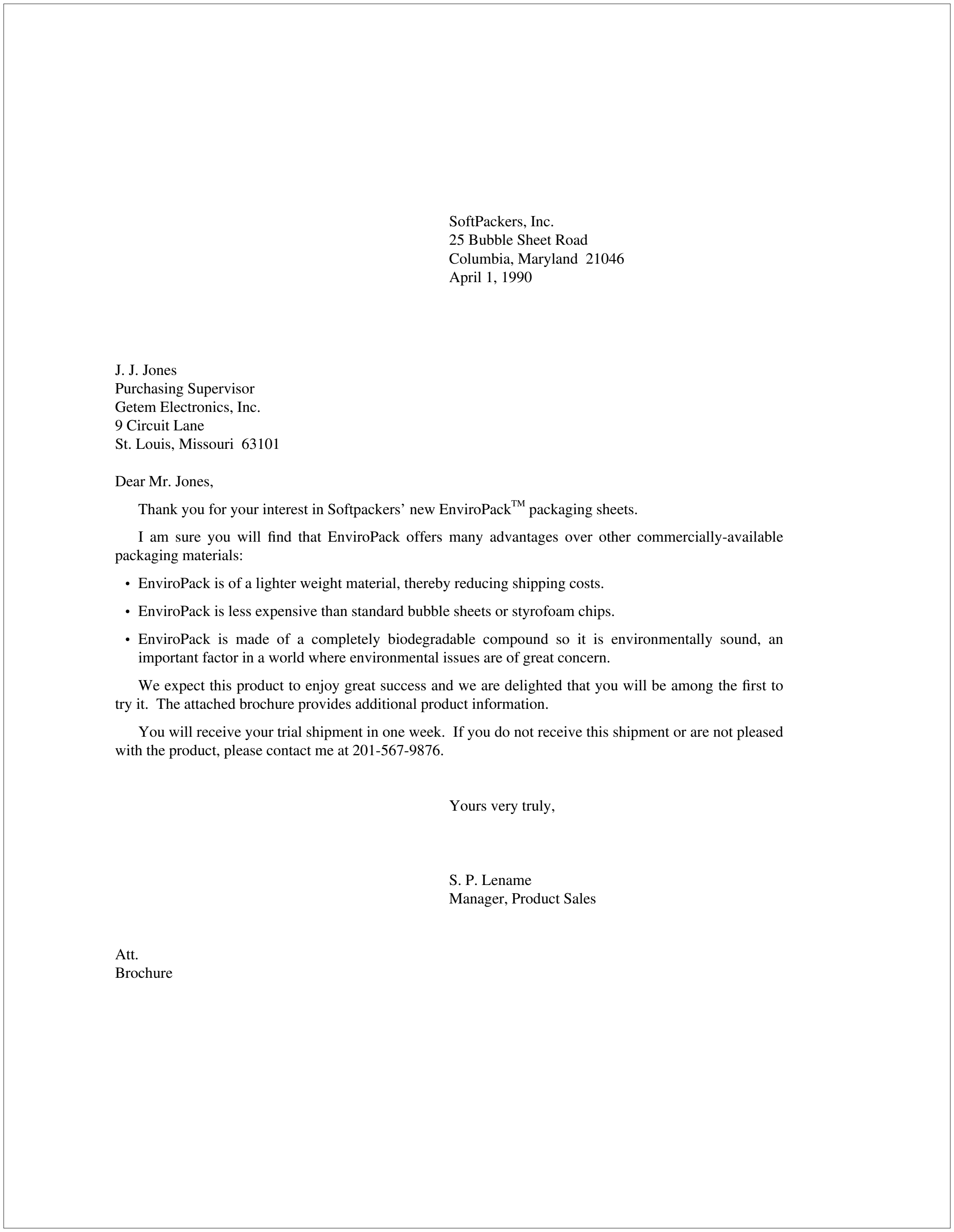
You will receive your trial shipment in one week. If you do not receive this shipment or are not pleased with the product, please contact me at 201-567-9876.

Yours very truly,

S. P. Lename
Manager, Product Sales

Att.
Brochure

OUTPUT: Semi-Blocked Business Letter (`.LT SB`)



SoftPackers, Inc.
25 Bubble Sheet Road
Columbia, Maryland 21046
April 1, 1990

J. J. Jones
Purchasing Supervisor
Getem Electronics, Inc.
9 Circuit Lane
St. Louis, Missouri 63101

Dear Mr. Jones,

Thank you for your interest in Softpackers' new EnviroPack^{TM} packaging sheets.

I am sure you will find that EnviroPack offers many advantages over other commercially-available packaging materials:

- EnviroPack is of a lighter weight material, thereby reducing shipping costs.
- EnviroPack is less expensive than standard bubble sheets or styrofoam chips.
- EnviroPack is made of a completely biodegradable compound so it is environmentally sound, an important factor in a world where environmental issues are of great concern.

We expect this product to enjoy great success and we are delighted that you will be among the first to try it. The attached brochure provides additional product information.

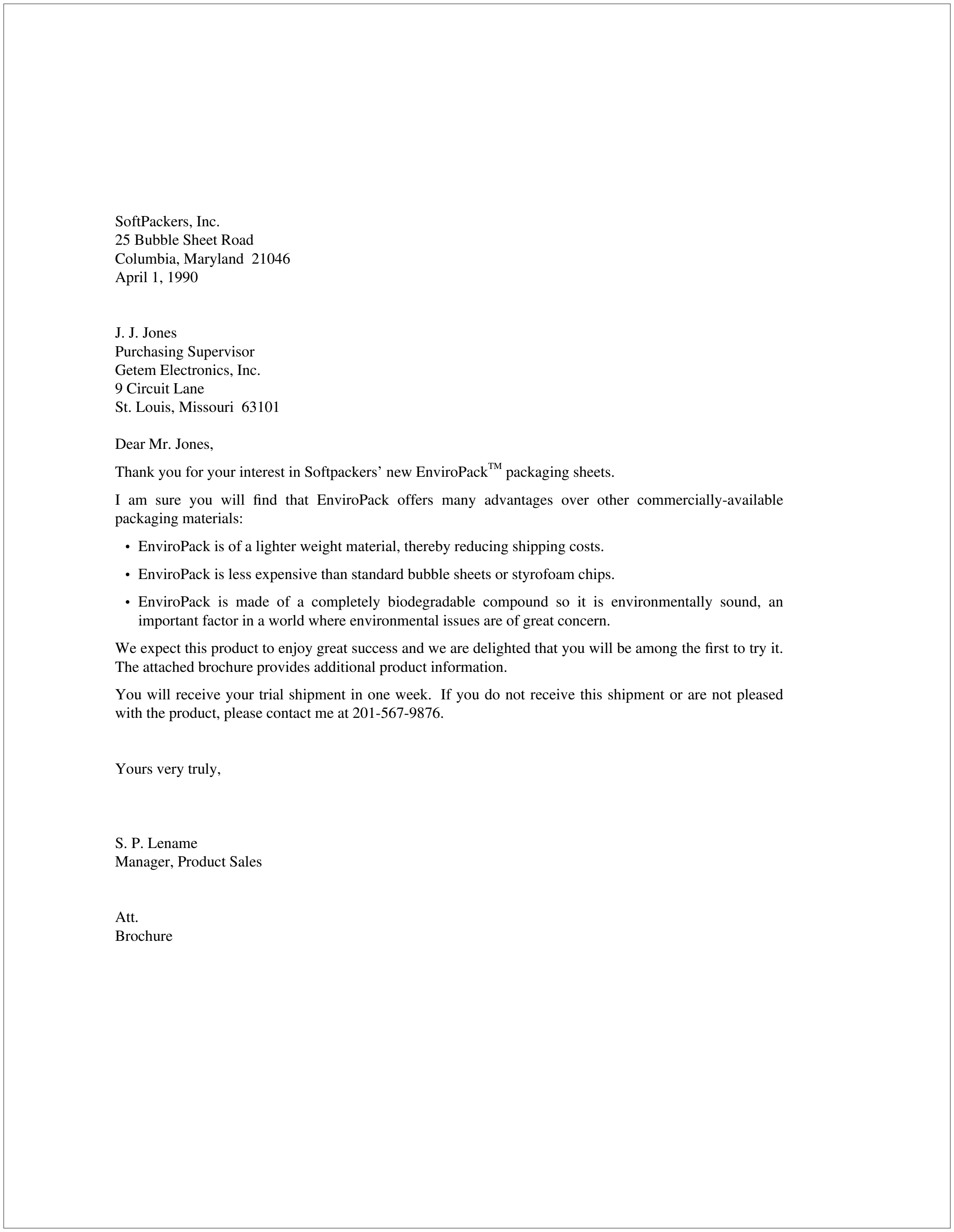
You will receive your trial shipment in one week. If you do not receive this shipment or are not pleased with the product, please contact me at 201-567-9876.

Yours very truly,

S. P. Lename
Manager, Product Sales

Att.
Brochure

OUTPUT: Full-Blocked Business Letter (`.LT FB`)



SoftPackers, Inc.
25 Bubble Sheet Road
Columbia, Maryland 21046
April 1, 1990

J. J. Jones
Purchasing Supervisor
Getem Electronics, Inc.
9 Circuit Lane
St. Louis, Missouri 63101

Dear Mr. Jones,

Thank you for your interest in Softpackers' new EnviroPack^{TM} packaging sheets.

I am sure you will find that EnviroPack offers many advantages over other commercially-available packaging materials:

- EnviroPack is of a lighter weight material, thereby reducing shipping costs.
- EnviroPack is less expensive than standard bubble sheets or styrofoam chips.
- EnviroPack is made of a completely biodegradable compound so it is environmentally sound, an important factor in a world where environmental issues are of great concern.

We expect this product to enjoy great success and we are delighted that you will be among the first to try it. The attached brochure provides additional product information.

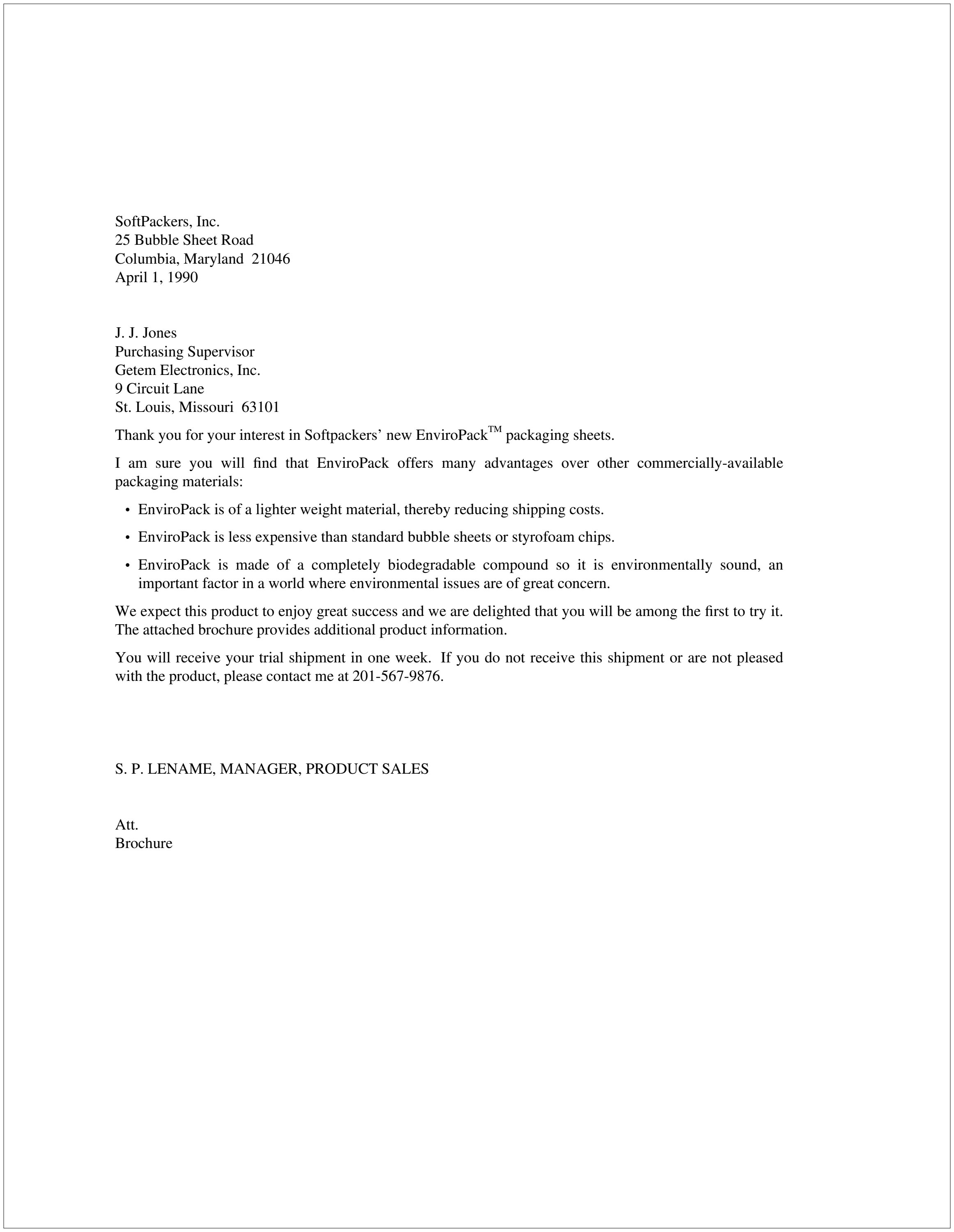
You will receive your trial shipment in one week. If you do not receive this shipment or are not pleased with the product, please contact me at 201-567-9876.

Yours very truly,

S. P. Lename
Manager, Product Sales

Att.
Brochure

OUTPUT: Simplified Business Letter (`.LT SP`)



SoftPackers, Inc.
25 Bubble Sheet Road
Columbia, Maryland 21046
April 1, 1990

J. J. Jones
Purchasing Supervisor
Getem Electronics, Inc.
9 Circuit Lane
St. Louis, Missouri 63101

Thank you for your interest in Softpackers' new EnviroPack™ packaging sheets.

I am sure you will find that EnviroPack offers many advantages over other commercially-available packaging materials:

- EnviroPack is of a lighter weight material, thereby reducing shipping costs.
- EnviroPack is less expensive than standard bubble sheets or styrofoam chips.
- EnviroPack is made of a completely biodegradable compound so it is environmentally sound, an important factor in a world where environmental issues are of great concern.

We expect this product to enjoy great success and we are delighted that you will be among the first to try it. The attached brochure provides additional product information.

You will receive your trial shipment in one week. If you do not receive this shipment or are not pleased with the product, please contact me at 201-567-9876.

S. P. LENAME, MANAGER, PRODUCT SALES

Att.
Brochure

APPENDIX E: Sample Footnotes

This example illustrates several footnote styles, using both labeled and automatically-numbered footnotes. The input on this page yields the output on the next page.

```
.FD 0
The first three footnotes are processed in the default footnote style (.FD 0).
This is an automatically-numbered footnote\*F
.FS
This is the first footnote example.
In this style (.FD 0), hyphenation is not permitted,
the right margin is justified, the footnote text is indented,
and the label is left-justified in the text-indent space.
.FE
followed by another one\*F
.FS
This is the second footnote example.
.FE
and a third one with a user-supplied label.***
.FS ***
This is the third footnote example.
The label (three asterisks) is supplied manually by the user.
.FE
.FD 11
The next two footnotes are processed in a different
footnote style (.FD 11).  Here is a labeled footnote\(dg\(dg
.FS \(dg\(dg
This is the fourth footnote example.
This footnote is labeled manually with two daggers.
In this style (.FD 11), hyphenation is permitted,
the right margin is not justified, the footnote text is indented,
and the footnote label is right-justified.
.FE
followed by an automatically-numbered footnote.\*F
.FS
This is the fifth footnote example.
This footnote is automatically-numbered.
Notice how the footnote label is right-justified with
the label of the previous footnote.
.FE
.FD 6
The last two footnotes are processed in another footnote style (.FD 6)
which generates a simple blocked format.
Here's an automatically-numbered footnote\*F
.FS
This is the sixth footnote example.
In this footnote style (.FD 6), hyphenation is not permitted,
the right margin is not jusitified, the text is not indented,
and the footnote label is left-justified.
.FE
and the last one is a labeled footnote.*****
.FS *****
This is the seventh and last footnote example.
This footnote has a very long label (five asterisks).
Of the seven footnotes on this page, four were numbered automatically
and three were labeled manually.
.FE
This concludes our footnote example.
```

Output:

The first three footnotes are processed in the default footnote style (.FD 0). This is an automatically-numbered footnote[1] followed by another one[2] and a third one with a user-supplied label.*** The next two footnotes are processed in a different footnote style (.FD 11). Here is a labeled footnote†† followed by an automatically-numbered footnote.[3] The last two footnotes are processed in another footnote style (.FD 6) which generates a simple blocked format. Here's an automatically-numbered footnote[4] and the last one is a labeled footnote.***** This concludes our footnote example.

1. This is the first footnote example. In this style (.FD 0), hyphenation is not permitted, the right margin is justified, the footnote text is indented, and the label is left-justified in the text-indent space.

2. This is the second footnote example.

*** This is the third footnote example. The label (three asterisks) is supplied manually by the user.

†† This is the fourth footnote example. This footnote is labeled manually with two daggers. In this style (.FD 11), hyphenation is permitted, the right margin is not justified, the footnote text is indented, and the footnote label is right-justified.

3. This is the fifth footnote example. This footnote is automatically-numbered. Notice how the footnote label is right-justified with the label of the previous footnote.

4. This is the sixth footnote example. In this footnote style (.FD 6), hyphenation is not permitted, the right margin is not jusitified, the text is not indented, and the footnote label is left-justified.

***** This is the seventh and last footnote example. This footnote has a very long label (five asterisks). Of the seven footnotes on this page, four were numbered automatically and three were labeled manually.

APPENDIX F: Error Messages

MM Error Messages

An MM error message has the following format:

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

where *filename* is the file in which the error occured, *n* is the line number in that file, and *description* is a descriptive message that indicates the macro in error and the nature of the erorr. Following is a list of descriptive messages (in alphabetical order by macro name) with some information on limitations or conditions that could cause that error.

Error Message	Explanation
`check TL, AU, AS, AE, MT sequence`	The order of macros at the beginning of a memorandum, released paper, or external letter is incorrect or some other macro has disturbed that order.
`check WA, WE, IA, IE, LT sequence`	The order of macros at the beginning of a business letter is incorrect or some other macro has disturbed that order.
`DE:no DS or DF active`	A display ends but has never started, i.e., a `.DE` macro occurs with no previous `.DS` or `.DF` macro to match it.
`DF:illegal inside TL or AS`	Floating displays are not allowed in the title or abstract.
`DF:missing DE`	A floating display starts inside a display, i.e., a previous `.DE` macro was omitted or mistyped.
`DF:missing FE`	A floating display starts inside a footnote. Most likely, a `.FE` macro to end a previous footnote was omitted or mistyped.
`DF:too many displays`	More than 26 floating displays are active at once, i.e., have accumulated but not yet output.
`DS:illegal inside TL or AS`	Static displays are not allowed in the title or abstract.
`DS:missing DE`	A static display starts inside a display, i.e., a previous `.DE` macro was omitted or mistyped.
`DS:missing FE`	A static display starts inside a footnote. Most likely, a `.FE` macro to end a previous footnote was omitted or mistyped.
`FE:no FS active`	A footnote ends but has never started, i.e., a `.FE` macro occurs with no previous `.FS` macro to match it.
`FS:missing DE`	A footnote starts inside a display. Most likely, a `.DE` macro to end a previous display was omitted or mistyped.
`FS:missing FE`	A footnote starts inside a footnote, i.e., a previous `.FS` macro was not matched by a closing `.FE` macro.
`H:bad arg:`*n*	The heading level argument must be a single digit from 1 to 7; *n* is not a legal value.
`H:missing arg`	The `.H` macro needs at least one argument.

Error Message	Explanation
`H:missing DE`	A heading macro (`.H` or `.HU`) occurs inside a display. Possibly, a `.DE` macro to end a previous display was omitted or mistyped.
`H:missing FE`	A heading macro (`.H` or `.HU`) occurs inside a footnote. Possibly, a `.FE` macro to end a previous footnote was omitted or mistyped.
`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 start when six lists are already active.
`LE:mismatched`	A list ends but has never started, i.e., a `.LE` macro occurs without a previous list initialization macro. Although this is not a fatal error, the message is issued because there almost certainly exists some problem in the preceding text.
`LI:no lists active`	A list item is requested but no list is active, i.e., a `.LI` macro occurs without a preceding list-initialization macro. Most likely, a list initialization macro was omitted, mistyped, or separated from the `.LI` macro by an intervening `.H` or `.HU` macro.
`LO argument not recognized`	The argument given to the `.LO` macro is incorrect; the only valid arguments are `SA`, `AT`, `SJ`, `RN`, and `CN`.
`LT argument not recognized`	The argument given to the `.LT` macro is incorrect; the only valid arguments are `BL`, `SB`, `FB`, and `SP`.
`ML:missing arg`	The `.ML` macro requires at least one argument.
`ND:missing arg`	The `.ND` macro requires one argument.
`RF:no RS active`	A reference ends but has never started, i.e., a `.RF` macro occurs with no previous `.RS` macro to match it.
`RP:missing RF`	A request to generate the reference page occurs inside a reference, i.e., a previous `.RS` macro was not matched by a closing `.RF` macro.
`RS:missing RF`	A reference starts inside a reference, i.e., a previous `.RS` macro was not matched by a closing `.RF` macro.
`S:bad arg` *n*	The `.S` macro was given an incorrect argument *n*.
`SA:bad arg:`*n*	The argument given to the `.SA` macro is incorrect; the only valid arguments are `0` or `1`.
`SG:missing DE`	The signature block is requested inside a display. Most likely, a `.DE` macro to end a previous display was omitted or mistyped.
`SG:missing FE`	The signature block is requested inside a footnote. Most likely, a `.FE` macro to end a previous footnote was omitted or mistyped.
`SG:no authors`	The `.SG` macro occurs without a previous `.AU` macro.

Error Message	Explanation
`TE: used TS H but no TH`	A table with repeating headers has ended but the header portion has not been defined. A table started with the `.TS H` macro must have a `.TH` macro to separate the repeating-header portion of the table from the data that follows.
`VL:missing arg`	The `.VL` macro requires at least one argument.
`WA macro missing`	The writer's address block has ended but it was never started, i.e., a `.WE` macro occurs with no previous `.WA` macro to match it.
`WA or WE macro missing`	The macros in the beginning of a business letter are not in order. The inside address block has started without a preceding writer's address block, i.e., a `.IA` macro occurs with no previous `.WA/.WE` macro pair.
`WA, WE, or IA macro missing`	The macros in the beginning of a business letter are not in order. The inside address block has ended but was never started or preceded by the writer's address block, i.e., a `.IE` macro occurs with no previous `.IA` macro to match it, or with no previous `.WA/.WE` macro pair.
`WA, WE, or IE macro missing`	The macros in the beginning of a business letter are not in order. The letter type macro cannot generate the letter without the writer's address block and the inside address block, i.e., the `.LT` macro occurs with no previous `.WA/.WE` and `.IA/.IE` macro pairs.
`WC:unknown option`	The `.WC` macro was given an incorrect argument.

Formatter Error Messages

Most messages issued by the *troff* text 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.

Error Message	Explanation
`cannot do ev`	This error is not an obvious one and occurs when some condition (e.g., setting a page length that is extremely short) does not allow a switch in environments.
`Cannot open file` *filename*	One of the files in the list of files to be processed cannot be opened, or the requested macro package does not exist.
`exception word list full`	Too many words have been specified in the hyphenation exception list (via `.hw` requests).
`Line overflow`	The output line being generated was too long for the internal line buffer. This excess was discarded. See the ‘‘`Word overflow`’’ message below.
`Can't open font file` *font*	A request has been made to mount or to use an unknown font. The description file for *font* cannot be found.
`Out of temp file space`	Additional temporary space for macro definitions, diversions, etc. cannot be allocated. This message often occurs because of unclosed diversions (missing `.FE` or `.DE` macros), unclosed macro definitions (e.g., missing ‘‘`..`’’), or a huge table of contents.
`too many page numbers`	The list of pages specified to the `–o` option is too long.
`too many number registers (`*n*`)`	The pool of number register names is full; the limit was exceeded by *n* names. Unneeded registers can be deleted with the `.rr` request.
`Too many (`*n*`) string/macro names`	The pool of string and macro names is full; the limit was exceeded by *n* names. Unneeded strings and macros can be deleted with the `.rm` request.
`Word overflow`	The word being generated exceeded an internal word buffer. The excess characters were discarded. A likely cause for this and for the ‘‘`Line Overflow`’’ message above are very long lines or words generated through the misuse of `\c` or of the `.cu` request, or very long equations produced by *eqn*. Word overflow can be cured by printing the offending word in no-fill mode (e.g., change an in-line equation to a displayed equation). The only cure for line overflow is to break the offending line into two lines.

APPENDIX G: Company-Related Information

This appendix summarizes company-related information in the MM package, such as the company logo, the company name, company location addresses, and proprietary markings and copyright notices.

Logo and Company Name

In memoranda, the company name and logo are used in the letterhead block on the first page and on the cover sheet, if any. In the released paper style, the company name follows the author's name(s) on the first page and on the cover sheet, if any. By default, the company name and logo are defined as:

Company name: AT&T Bell Laboratories

Logo: AT&T

The user-definable string `}Z` contains the company name, and the string `]S` contains the company logo. Use the `.AF` macro to change the company name (see §6.1.7.2).

Company Location Addresses

Location codes represent a particular company location or branch. In the MM package, location codes are assigned strings which contain the corresponding company address (city, state, and zip code). In the released-paper style, the location code for each author (given as the third argument to the `.AU` macro) generates the company address following the author's name(s) and affiliation on the first page and cover sheet, if any.

In some cases, branch locations share the same address. Here is an alphabetized list of recognized location codes and the corresponding company address:

Location Code	Company Address
`AK`	Norcross, Georgia 30071
`AL`	Allentown, Pennsylvania 18103
`ALC`	Allentown, Pennsylvania 18103
`AN`	Andover, Massachusetts 01810
`CB`	Columbus, Ohio 43213
`CH`	Chester, New Jersey 07930
`CP`	Piscataway, New Jersey 08854
`DR`	Denver, Colorado 80234
`FJ`	Holmdel, New Jersey 07733
`HL`	Short Hills, New Jersey 07078
`HK`	Short Hills, New Jersey 07078
`HO`	Holmdel, New Jersey 07733
`HOH`	Crawford Hill Laboratory Holmdel, New Jersey 07733
`HP`	South Plainfield, New Jersey 07080
`HR`	Middletown, New Jersey 07748
`IH`	Naperville, Illinois 60566
`IN`	Indianapolis, Indiana 46206
`INH`	Indianapolis, Indiana 46206
`IW`	Naperville, Illinois 60566
`IX`	Naperville, Illinois 60566
`LC`	Warren, New Jersey 07060
`LZ`	Lincroft, New Jersey 07738
`MH`	Murray Hill, New Jersey 07974
`MO`	Morristown, New Jersey 07960
`MT`	Middletown, New Jersey 07748
`MV`	North Andover, Massachusetts 01845
`PK`	Parsippany, New Jersey 07054
`PY`	Piscataway, New Jersey 08854
`RD`	Reading, Pennsylvania 19604
`RR`	Piscataway, New Jersey 08854
`SF`	Summit, New Jersey 07901
`SZ`	Springfield, New Jersey 07081
`WB`	West Long Branch, New Jersey 07764
`WH`	Whippany, New Jersey 07981
`WI`	Ward Hill, Massachusetts 01830
`WV`	Warren, New Jersey 07060

To define an address not on this list, see §6.1.8.

Proprietary Markings and Copyright Notice

Most companies use some form of proprietary marking or copyright notice to protect their assets and information. The wording for each company's markings and the types used vary.

The MM package provides six distinct markings, each with a specific purpose. The macro call

`.PM` [*type*] [*year*]

produces the desired marking at the bottom of the page:

`.PM P` or `.PM 1`

AT&T – PROPRIETARY
Use pursuant to Company Instructions.

`.PM RS` or `.PM 2`

AT&T – PROPRIETARY (RESTRICTED)
Solely for authorized persons having a need to know
pursuant to Company Instructions.

`.PM RG` or `.PM 3`

AT&T – PROPRIETARY (REGISTERED)
Solely for authorized persons having a need to know
and subject to cover sheet instructions.

`.PM CP` or `.PM 4`

SEE PROPRIETARY NOTICE ON COVER PAGE

`.PM CR` or `.PM 5`

Copyright © 1990 AT&T
All Rights Reserved.

`.PM UW` or `.PM 6`

THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF
AT&T AND IS NOT TO BE DISCLOSED OR USED EXCEPT IN
ACCORDANCE WITH APPLICABLE AGREEMENTS.

Copyright © 1990 AT&T
Unpublished & Not for Publication
All Rights Reserved.

By default, the current year is used for those markings that include a copyright notice. To override the default, give the year as the second argument to the `.PM` macro.

APPENDIX H: Summary of Macros, Strings, and Number Registers

Macros

The following is an alphabetical list of macro names used by MM. For each macro entry, the first line gives the name of the macro, a brief description, and a reference to the section in which the macro is described. The second line gives a synopsis of the macro call.

Macros whose descriptions are followed by † are not, in general, invoked directly by the user. Rather, they are ‘‘user exit’’ macros that are defined by the user and called from inside header, footer, or other macros.

Macro	Description and Synopsis	Section
`1C`	One-column processing `.1C`	12.3
`2C`	Two-column processing `.2C`	12.3
`AE`	Abstract end `.AE`	6.1.4
`AF`	Alternate First-Page format `.AF` [*company-name*]	6.1.7.2, Appendix G
`AL`	Automatically-numbered List start `.AL` [*type*] [*text-indent*] [`1`]	5.3.3.1
`AS`	Abstract start `.AS` [*mode*] [*indent*] [*type*]	6.1.4
`AT`	Author’s Title `.AT` [*title*] …	6.1.2
`AU`	Author information `.AU` *name* [*initials*] [*location*] [*organization*] [*phone*] [*room*] [*arg*] [*arg*] [*arg*]	6.1.2
`AV`	Approval Signature `.AV` [*approver’s-name*] [`1`]	6.3.3
`B`	Bold `.B` [*bold*] [*previous*] …	12.1
`BE`	Bottom Block end `.BE`	9.6
`BI`	Bold/Italic `.BI` [*bold*] [*italic*] …	12.1
`BL`	Bullet List start `.BL` [*text-indent*] [`1`]	5.3.3.2
`BR`	Bold/Roman `.BR` [*bold*] [*roman*] …	12.1
`BS`	Bottom Block start `.BS`	9.6
`CS`	Cover Sheet `.CS`	10.2

Macro	Description and Synopsis	Section
`DE`	Display end `.DE`	7.1, 7.2
`DF`	Floating Display start `.DF` [*format*] [*fill*] [*right-indent*]	7.2
`DL`	Dash List start `.DL` [*text-indent*] [`1`]	5.3.3.3
`DS`	Static Display start `.DS` [*format*] [*fill*] [*right-indent*]	7.1
`EC`	Equation Caption `.EC` [*title*] [*override*] [*flag*]	7.7
`EF`	Even-page Footer `.EF` [*footer*]	9.3.1
`EH`	Even-page Header `.EH` [*header*]	9.2.1
`EN`	Equation end `.EN`	7.4
`EQ`	Equation start `.EQ` [*label*]	7.4
`EX`	Exhibit Caption `.EX` [*title*] [*override*] [*flag*]	7.7
`FC`	Formal Closing `.FC` [*closing*]	6.3.1.1
`FD`	Footnote Default format `.FD` [*mode*] [`1`]	8.4, Appendix E
`FE`	Footnote end `.FE`	8, Appendix E
`FG`	Figure Title `.FG` [*title*] [*override*] [*flag*]	7.7
`FS`	Footnote start `.FS` [*label*]	8, Appendix E
`G1`	Graph start `.G1`	7.6
`G2`	Graph end `.G2`	7.6
`H`	Heading — Numbered `.H` *level* [*heading-text*] [*heading-suffix*]	4.2
`HC`	Hyphenation Character `.HC` [*hyphenation-indicator*]	3.4
`HM`	Heading Mark numbering style (Arabic or Roman numerals, or letters) `.HM` [*style1*] … [*style7*]	4.2.2.6

Macro	Description and Synopsis	Section
`HU`	Heading — Unnumbered `.HU` *heading-text*	4.3
`HX`	Heading — User-exit X macro (before heading is printed) † `.HX` *dlevel rlevel heading-text*	4.6
`HY`	Heading — User-exit Y macro (before heading is printed) † `.HY` *dlevel rlevel heading-text*	4.6
`HZ`	Heading — User-exit Z macro (after heading is printed) † `.HZ` *dlevel rlevel heading-text*	4.6
`I`	Italic `.I` [*italic*] [*previous*] …	12.1
`IA`	Inside Address start `.IA` [*name* [*title*]]	6.2.3
`IB`	Italic/Bold `.IB` [*italic*] [*bold*] …	12.1
`IE`	Inside Address end `.IE`	6.2.3
`IR`	Italic/Roman `.IR` [*italic*] [*roman*] …	12.1
`LB`	List Begin `.LB` *text-indent mark-indent pad type* [*mark*] [*LI-space*] [*LB-space*]	5.4
`LC`	List Clear `.LC` [*list-level*]	Appendix B
`LE`	List end `.LE [1]`	5.3.2
`LI`	List Item `.LI` [*mark*] `[1]`	5.3.1
`LO`	Letter Options `.LO` *type* [*notation*]	6.2.4
`LT`	Letter Types `.LT` [*type*]	6.2.1, Appendix D
`ML`	Marked List start `.ML` *mark* [*text-indent*] `[1]`	5.3.3.4
`MT`	Memorandum Types `.MT` [*type*] [*addressee*] or `.MT 4 [1]`	6.1.6, 6.1.8, Appendix C
`ND`	New Date `.ND` *new-date*	6.1.7.1
`NE`	Notation end `.NE`	6.3.2
`NS`	Notation start `.NS` [*type*] `[1]`	6.3.2

Macro	Description and Synopsis	Section
`nP`	Numbered Paragraph with Double-Line Indent `.nP`	4.1
`OF`	Odd-page Footer `.OF` [*footer*]	9.3.2
`OH`	Odd-page Header `.OH` [*header*]	9.2.2
`OK`	Other Keywords `.OK` [*keyword*] …	6.1.5
`OP`	Odd-numbered Page `.OP`	12.6
`P`	Paragraph `.P` [*type*]	4.1
`PS`	Picture start `.PS`	7.5
`PF`	Page Footer `.PF` [*footer*]	9.3
`PH`	Page Header `.PH` [*header*]	9.2
`PM`	Proprietary Marking `.PM` [*type*] [*year*]	9.7, Appendix G
`PE`	Picture end `.PE`	7.5
`PX`	Page Header — User-exit macro † `.PX`	9.5
`R`	Return to Roman (regular) font `.R`	12.1
`RB`	Roman/Bold `.RB` [*roman*] [*bold*] …	12.1
`RD`	Read Insertion `.RD` [*prompt*] [*diversion-name*] [*string-name*]	12.10
`RF`	Reference end `.RF`	11
`RI`	Roman/Italic `.RI` [*roman*] [*italic*] …	12.1
`RL`	Reference List start `.RL` [*text-indent*] [`1`]	5.3.3.5
`RP`	Reference Page `.RP` [*arg*] [*arg*]	11.3
`RS`	Reference start `.RS` [*string-name*]	11

Macro	Description and Synopsis	Section
`S`	Set Point Size and Vertical Spacing `.S` [*point-size*] [*vertical-spacing*]	12.7
`SA`	Set Adjustment default (right-margin justification) `.SA` [*mode*]	12.2
`SG`	Signature line `.SG` [*initials*] [`1`]	6.3.1.2
`SK`	Skip Pages `.SK` [*pages*]	12.5
`SM`	Smaller (one-point size reduction) `.SM` *string1* [*string2*] [*string3*]	12.8
`SP`	Blank Vertical Spaces `.SP` [*lines*]	12.4
`TB`	Table Title `.TB` [*title*] [*override*] [*flag*]	7.7
`TC`	Table of Contents `.TC` [*slevel*] [*spacing*] [*tlevel*] [*tab*] [*head1*] [*head2*] [*head3*] [*head4*] [*head5*]	10.1
`TE`	Table end `.TE`	7.3
`TH`	Table Repeating-Header `.TH [N]`	7.3
`TL`	Title `.TL` [*charging-case*] [*filing-case*]	6.1.1
`TM`	Document Identification Number `.TM` [*number*] …	6.1.3
`TP`	Top-of-Page macro † `.TP`	9.5
`TS`	Table start `.TS [H]`	7.3
`TX`	Table of Contents — User-exit macro † `.TX`	10.1
`TY`	Table of Contents — User-exit macro (supress ‘‘CONTENTS’’) † `.TY`	10.1
`VL`	Variable-item List start `.VL` *text-indent* [*mark-indent*] [`1`]	5.3.3.6
`WA`	Writer’s Address start `.WA` *name* [*title*]	6.2.2
`WC`	Width Control `.WC` [*code*]	12.3.1
`WE`	Writer’s Address end `.WE`	6.2.2

Strings

The following is an alphabetical list of string names used by MM. For each string entry, the first line gives the name of the string, a brief description, and a reference to the section in which the string is described. The second line gives the initial (default) definition.

String	Description and Initial Definition	Section
`BU`	Bullet (•) symbol `\s-2\(bu\s0`	3.7.1
`Ci`	Table of Contents Indent list (for heading levels 1 through 7) `0 0 0 0 0 0 0`	10.1
`DT`	Date (override with `.ND`) Current date: *month day, year* (e.g., `April 1, 1990`)	1.4, 6.1.7.1
`EM`	Em dash (—) `\(em`	3.7.2
`F`	Footnote mark `\v'-.4m'\s-3\\n+(:p\s0\v'.4m'`	8.1
`HF`	Heading Font list (for heading levels 1 through 7) `3 3 2 2 2 2 2`	4.2.2.4
`HP`	Heading Point Size list (for heading levels 1 through 7) `-1 -1 0 0 0 0 0`	4.2.2.5
`Le`	Title for ‘‘List of Equations’’ Page `LIST OF EQUATIONS`	7.8
`Lf`	Title for ‘‘List of Figures’’ Page `LIST OF FIGURES`	7.8
`Lt`	Title for ‘‘List of Tables’’ Page `LIST OF TABLES`	7.8
`Lx`	Title for ‘‘List of Exhibits’’ Page `LIST OF EXHIBITS`	7.8
`Rf`	Reference mark `\v'-.4m'\s-3[\\n+(:R]\s0\v'.4m'`	11.1
`Rg`	Registered Mark symbol ® `\v'-0.4m'\s-4\(rg\s+4\v'0.4m'`	3.7.3
`Rp`	Title for Reference Page `REFERENCES`	11.3
`Sm`	Service Mark symbol SM `\v'-0.5m'\s-4SM\s+4\v'0.5m'`	3.7.3
`Tm`	Trademark symbol TM `\v'-0.5m'\s-4TM\s+4\v'0.5m'`	3.7.3
`]S`	Logo (for use on letterhead) `\s36\(LH\s0`	6.1.7.2, Appendix G
`}Z`	Company name `AT&T Bell Laboratories`	6.1.7.2, Appendix G

See §1.4 for information on how to define and reference strings. See §12.9 for the accent strings.

Number Registers

The following is an alphabetical list of number register names used by MM. For each number register entry, the first line gives the name of the register, a brief description, and a reference to the section in which the register is described. The second line gives the initial (default) value and the legal values. The legal values are enclosed in square brackets [ ] and may be a comma-separated list of values (i.e., [*n*, *n*, *n*]) or a range of values (i.e., [*m*-*n*] meaning *m* to *n* inclusive; if *n* is `?`, then there is no upper limit).

Any single-character register name can be set from the command line (see §2.3). Number registers whose descriptions are followed by † must be set before the MM macro package is intialized (see §2.3 and §2.4).

See §1.4 for information on how to set number registers.

Register	Description, and Initial and Legal Values	Section
`A`	Alternate first-page format † `0    [0,1]`	2.3, Appendix A
`Au`	Inhibit author's company information in ‘‘from’’ block `1    [0,1]`	6.1.2
`C`	Copy type (e.g., DRAFT) † `0    [0-5]`	2.3
`Cl`	Table of Contents level of headings saved `2    [0-7]`	4.4, 10.1
`Cp`	Placement of list of figures, tables, equations, and exhibits `1    [0,1]` (e.g., on separate pages)	7.8, 10.1
`D`	Debug flag † `0    [0,1]`	2.3
`De`	Floating display eject `0    [0,1]`	7.2
`Df`	Floating display format `5    [0-5]`	7.2
`Ds`	Static display pre-space and post-space `1    [0,1]`	7.1
`E`	Font of ‘‘subject/date/from’’ fields † `1    [0,1]`	2.3
`Ec`	Equation counter (incremented and used by `.EC` macro) `0    [0-?]`	7.7
`Ej`	Page eject for heading level `0    [0-7]`	4.2.2.1
`Eq`	Equation Label placement (e.g., right-adjusted) `0    [0,1]`	7.4
`Ex`	Exhibit counter (incremented and used by `.EX` macro) `0    [0-?]`	7.7
`Fg`	Figure counter (increment and used by `.FG` macro) `0    [0-?]`	7.7

Register	Description, and Initial and Legal Values	Section
`Fs`	Footnote space (i.e., spacing between footnotes) `1 [0-?]`	8.3
`H1`	Heading counter for level-1 headings `0 [0-?]`	4.2.2
`H2`	Heading counter for level-2 headings `0 [0-?]`	4.2.2
`H3`	Heading counter for level-3 headings `0 [0-?]`	4.2.2
`H4`	Heading counter for level-4 headings `0 [0-?]`	4.2.2
`H5`	Heading counter for level-5 headings `0 [0-?]`	4.2.2
`H6`	Heading counter for level-6 headings `0 [0-?]`	4.2.2
`H7`	Heading counter for level-7 headings `0 [0-?]`	4.2.2
`Hb`	Heading break level (after `.H` and `.HU` macros) `2 [0-7]`	4.2.2.2
`Hc`	Heading centering level (for `.H` and `.HU` macros) `0 [0-7]`	4.2.2.3
`Hi`	Heading temporary indent (after `.H` and `.HU` macros) `1 [0-2]`	4.2.2.2
`Hs`	Heading space level (after `.H` and `.HU` macros) `2 [0-7]`	4.2.2.2
`Ht`	Heading type (for `.H` macro: single or concatenated numbers) `0 [0,1]`	4.2.2.6
`Hu`	Heading level for unnumbered heading (`.HU` macro) `2 [0-7]`	4.3
`Hy`	Hyphenation control `0 [0,1]`	3.4
`L`	Page length † `11i [2i-?]`	2.3, Appendix A
`Le`	List of Equations `0 [0,1]`	7.8
`Lf`	List of Figures `0 [0,1]`	7.8
`Li`	List indent for automatically-numbered lists `5 [0-?]`	5.3.3.1, Appendix A
`Ls`	List spacing between items by level `6 [0-6]`	5.1

Register	Description, and Initial and Legal Values	Section
`Lt`	List of Tables `0 [0,1]`	7.8
`Lx`	List of Exhibits `0 [0,1]`	7.8
`N`	Page numbering style † `0 [0-5]`	2.3
`Np`	Numbering style for paragraphs (unnumbered or numbered) `0 [0,1]`	4.1
`O`	Page offset (i.e., left margin) † `1i [0i-?]`	2.3, Appendix A
`Oc`	Numbering style for Table of Contents page (e.g., lower-case Roman numerals) `0 [0,1]`	10.1
`Of`	Figure caption style (e.g., period separator) `0 [0,1]`	7.7
`P`	Page number (managed by the MM package) † `0 [0-?]`	2.3, 9.4
`Pi`	Paragraph indent `3 [0-?]`	4.1, Appendix A
`Ps`	Paragraph spacing (e.g., one blank vertical space) `1 [0-?]`	4.1
`Pt`	Paragraph type (e.g., blocked, indented) `0 [0-2]`	4.1
`Pv`	‘‘PRIVATE’’ header (e.g., not printed, printed) `0 [0-2]`	9.8
`S`	Point Size † `10 [6-?]`	2.3, Appendix A
`Si`	Standard indent for displays `3 [0-?]`	7.1, 7.2, Appendix A
`T`	Terminal type † `0 [0-2]`	Appendix A
`Tb`	Table counter (incremented and used by `.TB` macro) `0 [0-?]`	7.7
`U`	Underlining style (e.g., continuous) † `0 [0,1]`	Appendix A
`W`	Page width (e.g., line length) `6i [2i-?]`	2.3, Appendix A
`:R`	Reference counter (used by string `Rf` and `.RS` macro) `0 [0-?]`	11.1
`:p`	Footnote counter (used by string `F` and `.FS` macro) `0 [0-?]`	8.1