1098

PWB/MM

Programmer's Workbench

Memorandum Macros

D. W. Smith
J. R. Mashey
E. C. Pariser (January 1980 Reissue)

Bell Telephone Laboratories, Incorporated

PWB/MM
Programmer's Workbench Memorandum Macros

LIST OF FIGURES

PWB/MM—Programmer's Workbench Memorandum Macros

D. W. Smith
J. R. Mashey
E. C. Pariser (January 1980 Reissue)

Bell Laboratories
Piscataway, New Jersey 08854

1. INTRODUCTION

1.1 Purpose

This memorandum is the user's guide and reference manual for PWB/MM (or just **-mm**), a general-purpose package of text formatting macros for use with the UNIX† text formatters *nroff* [9] and *troff* [9]. The purpose of PWB/MM is to provide to the users of PWB/UNIX a unified, consistent, and flexible tool for producing many common types of documents. Although PWB/UNIX provides other macro packages for various *specialized* formats, PWB/MM has become the standard, general-purpose macro package for most documents.

PWB/MM can be used to produce:

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

The uses of PWB/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 PWB/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 PWB/MM defaults are acceptable. Likewise, each section progresses from normal-case to special-case facilities. We recommend reading a section in detail only until there is enough information to obtain the desired format, then skimming the rest of it, because some details may be of use to just a few people.

Numbers enclosed in curly brackets ({}) refer to section numbers within this document. For example, this is {1.2}.

Sections that require knowledge of the formatters {1.4} have a bullet (•) at the end of the section heading.

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.

A reference of the form *name*(N) points to page *name* in section *N* of the *PWB/UNIX User's Manual* [1].

---

† UNIX is a Trademark of Bell Laboratories.

The examples of *output* in this manual are as produced by *troff*; *nroff* output would, of course, look somewhat different (Appendix D shows *both* the *nroff* and *troff* output for a simple letter). In those cases in which the behavior of the two formatters is truly different, the *nroff* action is described first, with the *troff* action following in parentheses. For example:

The title is underlined (bold).

means that the title is underlined in *nroff* and bold in *troff*.

1.3 Overall Structure of a Document

The input for a document that is to be formatted with PWB/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 [9], 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 name, 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 [4]. 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 [5]. The body may also contain various types of displays, tables, figures, and footnotes [7, 8].
- *Ending*—This segment contains those items that occur once only, at the end of a document. Included here are signature(s) and lists of notations (e.g., "copy to" lists) [6.12]. 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 [10].

The existence and size of these four segments varies widely among different document types. Although a specific item (such as date, title, author name(s), 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 *nroff* and *troff*.

Requests are built-in commands recognized by the formatters. Although one seldom needs to use these requests directly [3.9], this document contains references to some of them. Full details are given in [9]. For example, the request:

```
.sp
```

inserts a 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. PWB/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 1-character names) or "*(" (for 2-character names). For instance, the string *DT* in PWB/MM normally contains the current date, so that the *input* line:

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

may result in the following *output:*

```
Today is January 22, 1980.
```

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

```
.ds DT 01/01/79
```

or by invoking a macro designed for that purpose {6.7.1}.

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 1-character names) or "\n(" (for 2-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 {13.1} regarding naming conventions for requests, macros, strings, and number registers.

1.5 Prerequisites and Further Reading

1.5.1 Prerequisites. We assume familiarity with UNIX at the level given in [3] and [4]. Some familiarity with the request summary in [9] is helpful.

1.5.2 Further Reading. [9] provides detailed descriptions of formatter capabilities, while [5] provides a general overview. See [6] (and possibly [7]) for instructions on formatting mathematical expressions. See *tbl*(1) and [11] for instructions on formatting tabular data.

Examples of formatted documents and of their respective input, as well as a quick reference to the material in this manual are given in [8].

2. INVOKING THE MACROS

This section tells how to access PWB/MM, shows PWB/UNIX command lines appropriate for various output devices, and describes command-line flags for PWB/MM. Note that file names, program names, and typical command sequences apply only to PWB/UNIX; different names and command lines may have to be used on other systems.

2.1 The mm Command

The *mm*(1) command can be used to print documents using *nroff* and PWB/MM; this command invokes *nroff* with the -mm flag {2.2}. It has options to specify preprocessing by *tbl*(1) and/or by *neqn*(1), and for postprocessing by various output filters. Any arguments or flags that are not recognized by *mm*(1), e.g. -rC3, are passed to *nroff* or to PWB/MM, as appropriate. The options, which can occur in any order but *must* appear before the file names, are:

-e	*neqn*(1) is to be invoked.
-t	*tbl*(1) is to be invoked.
-c	*col*(1) is to be invoked.
-E	the "-e" option of *nroff* is to be invoked.
-12	need 12-pitch mode. Be sure that the pitch switch on the terminal is set to 12.
-T300	output is to a DASI300 terminal. This is the *default* terminal type (unless $TERM is set).
-T300-12	output is to a DASI300 in 12-pitch mode.
-T300s	output is to a DASI300S.
-T300S	output is to a DASI300S.
-T300s-12	output is to a DASI300S in 12-pitch mode.
-T300S-12	output is to a DASI300S in 12-pitch mode.
-T4014	output is to a Tektronix 4014.
-Thp	output is to a HP264x.

-T450 output is to a DASI450.
-T450-12 output is to a DASI450 in 12-pitch mode.
-Ttn output is to a GE TermiNet 300.
-Ttn300 output is to a GE TermiNet 300.
-Tti output is to a Texas Instrument 700 series terminal.
-T37 output is to a TELETYPE® Model 37.
-T43 output is to a TELETYPE® Model 43.

2.2 The -mm Flag

The PWB/MM package can also be invoked by including the **-mm** flag as an argument to the formatter. It causes the file /usr/lib/tmac/tmac.m to be read and processed before any other files. This action defines the PWB/MM macros, sets default values for various parameters, and initializes the formatter to be ready to process the files of input text.

2.3 Typical Command Lines

The prototype command lines are as follows (with the various options explained in [2.4] and in [9]).

- Text without tables or equations:

  mm [options] filename ...
  or **nroff** [options] **-mm** filename ...
  or **troff** [options] **-mm** filename ...

- Text with tables:

  mm -t [options] filename ...
  or **tbl** filename ... | **nroff** [options] **-mm**
  or **tbl** filename ... | **troff** [options] **-mm**

- Text with equations:

  mm -e [options] filename ...
  or **neqn** filename ... | **nroff** [options] **-mm**
  or **eqn** filename ... | **troff** [options] **-mm**

- Text with both tables and equations:

  mm -t -e [options] filename ...
  or **tbl** filename ... | **neqn** | **nroff** [options] **-mm**
  or **tbl** filename ... | **eqn** | **troff** [options] **-mm**

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. Some commonly-used terminal types and the command lines appropriate for them are given below. See [2.4] as well as *300*(1), *450*(1), *hp*(1), *col*(1), and *terminals*(7) for further information.

- DASI300 (GSI300/DTC300) in 10-pitch, 6 lines/inch mode and a line length of 65 characters:

  mm filename ...
  or **nroff -T300 -h -mm** filename ...

- DASI300 (GSI300/DTC300) in 12-pitch, 6 lines/inch mode and a line length of 80—rather than 65—characters:

  mm -12 filename ...
  or **nroff -T300-12 -rW80 -rO3 -h -mm** filename ...

  or, equivalently (and more succinctly):

nroff -T300-12 -rT1 -h -mm filename ...

- DASI450 in 10-pitch, 6 lines/inch mode:

  mm -T450 filename ...
  or **nroff -T450 -h -mm** filename ...

- DASI450 in 12-pitch, 6 lines/inch mode:

  mm -T450-12 filename ...
  or **nroff -T450-12 -rW80 -rO3 -h -mm** filename ...
  or **nroff -T450-12 -rT1 -h -mm** filename ...

- Hewlett-Packard HP264x CRT family:

  mm -Thp filename ...
  or **nroff -h -mm** filename ... **| hp**

- Any terminal incapable of reverse paper motion (GE TermiNet, Texas Instruments 700 series, etc.):

  mm -Ttn filename ...
  or **nroff -mm** filename ... **| col**

- Versatec printer (see *vp*(1) for additional details):

  vp [vp-options] **"mm -rT2 -c** filename ..."
  or **vp** [vp-options] **"nroff -rT2 -mm** filename ... **| col"**

Of course, *tbl*(1) and *eqn*(1)/*neqn*(1), if needed, must be invoked as shown in the command line prototypes at the beginning of this section.

If two-column processing {11.4} is used with *nroff*, either the -c option must be specified to *mm*(1), or the *nroff* output must be postprocessed by *col*(1). In the latter case, the -T37 terminal type must be specified to *nroff*, the -h option must *not* be specified, and the output of *col*(1) must be processed by the appropriate terminal filter (e.g., *300*(1)); *mm*(1) with the -c option handles all this automatically.

2.4 Parameters that Can Be Set from the Command Line

Number registers are commonly used within PWB/MM to hold parameter values that control various aspects of output style. Many of these can be changed within the text files via .nr requests. In addition, some of these 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*—see below) *must* be set on the command line (or before the PWB/MM macro definitions are processed) and their meanings are:

-rA1 has the effect of invoking the .AF macro without an argument {6.7.2}.

-r*B*n defines the macros for the cover sheet and the table of contents. If *n* is 1, table-of-contents processing is enabled. If *n* is 2, then cover-sheet processing will occur. If *n* is 3, both will occur. That is, *B* having a value greater than 0 *defines* the .TC {10.1} and/or .CS {10.2} macros. Note that to have any effect, these macros must also be *invoked.*

-r*C*n *n* sets the type of copy (e.g., DRAFT) to be printed at the bottom of each page. See {9.5}.
n = 1 for OFFICIAL FILE COPY.
n = 2 for DATE FILE COPY.
n = 3 for DRAFT.

-rD1 sets *debug mode*. This flag requests the formatter to attempt to continue processing even if PWB/MM detects errors that would otherwise cause termination. It also includes some debugging information in the default page header {9.2, 11.3}.

-rL*k* sets the length of the physical page to *k* lines.[1] The default value is 66 lines per page. This parameter is used for obtaining 8 lines-per-inch output on 12-pitch terminals, or when directing output to a Versatec printer.

-rN*n* specifies the page numbering style. When *n* is 0 (default), all pages get the (prevailing) header {9.2}. When *n* is 1, the page header replaces the footer on page 1 only. When *n* is 2, the page header is omitted from page 1. When *n* is 3, "section-page" numbering {4.5} occurs. When *n* is 4, the *default* page header is suppressed; however a user-specified header is not affected.

n	*Page 1*	*Pages 2 ff.*
0	header	header
1	header replaces footer	header
2	no header	header
3	"section-page" as *footer*	
4	no header	no header unless PH defined

The contents of the prevailing header and footer do *not* depend on of the value of the number register *N*; *N* only controls whether and where the header (and, for *N*= 3, the footer) is printed, as well as the page numbering style. In particular, if the header and footer are null {9.2, 9.5}, the value of *N* is irrelevant.

-rO*k* offsets output *k* spaces to the right.[1] It is helpful for adjusting output positioning on some terminals. NOTE: 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. The default *n* is 10, i.e., 10-point type on 12-point leading (vertical spacing), giving 6 lines per inch {11.8}. This parameter applies to *troff* only.

-rT*n* provides register settings for certain devices. If *n* is 1, then the line length and page offset are set for output directed to a DASI300 or DASI450 in 12-pitch, 6 lines/inch mode, i.e., they are set to 80 and 3, respectively. Setting *n* to 2 changes the page length to 84 lines per page and inhibits underlining; it is meant for output sent to the Versatec printer. The default value for *n* is 0. This parameter applies to *nroff* only.

-rU1 controls underlining of section headings. This flag causes only letters and digits to be underlined. Otherwise, all characters (including spaces) are underlined {4.2.2.4.2}. This parameter applies to *nroff* only.

-rW*k* page width (i.e., line length and title length) is set to *k*.[1] This can be used to change the page width from the default value of 65 characters (6.5 inches).

2.5 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:

```
zero or more initializations of registers listed in {2.4}
.so /usr/lib/tmac/tmac.m
remainder of text
```

1. For *nroff*, *k* is an *unscaled* number representing lines or character positions; for *troff*, *k* must be *scaled*.

In this case, one must *not* use the **-mm** flag (nor the *mm*(1) command); the .so request has the equivalent effect, but the registers in {2.4} 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 W 80
.nr O 10
.nr N 3
.nr B 1
.so /usr/lib/tmac/tmac.m
.H 1 "INTRODUCTION"
   :
   :
```

specifies, for *nroff*, a line length of 80, a page offset of 10, "section-page" numbering, and table of contents processing.

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 are hyphenated {3.4} as necessary. It is possible to turn any of these modes on and off (see .SA {11.2}, *Hy* {3.4}, and the formatter .nf and .fi requests [9]). 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 PWB/MM macros cause a break.

While formatter requests can be used with PWB/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 facilities (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 {3.9}.

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 in {6.6}). 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 (").[2] Otherwise, it will be treated as several separate arguments.

---

2. A double quote (") is a *single* character that must not be confused with two apostrophes or acute accents (''), or with two grave accents (``).

Double quotes (") are *not* permitted as part of the value of a macro argument or of a string that is to be used as a macro argument. If 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 type a backslash followed by a space ("\ "). This pair of characters directly generates an *unpaddable space*. Second, one may sacrifice some seldom-used character to be translated into a 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 ~
```

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

```
.tr ~~
```

before the place where it is needed. 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. Note that the use of the tilde in this fashion is *not* recommended for documents in which the tilde is used within equations.

3.4 Hyphenation

The formatters (and, therefore, PWB/MM) will automatically hyphenate words, if need be. However, the user may specify the hyphenation points for a specific occurrence of any word by the use of a special character known as a hyphenation indicator, or may specify hyphenation points for a small list of words (about 128 characters).

If the *hyphenation indicator* (initially, the two-character sequence "\%") appears at the beginning of a word, the word is *not* hyphenated. Alternatively, it can be used to indicate legal hyphenation point(s) 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 (^) is often used for this purpose; this is done by inserting the following at the beginning of a document:

```
.HC ^
```

Note that 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.*

Hyphenation can be turned off in the body of the text by specifying:

```
.nr Hy 0
```

once at the beginning of the document. For hyphenation control within footnote text and across pages, see {8.3}.

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 {6.6}, .TC {10.1}, and .CS {10.2} use the formatter .ta request to set tab stops, and then restore the *default* values[3] of tab settings. Thus, setting tabs to other than the default values is the user's responsibility.

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 {3.1}.

Also note that *tbl*(1) {7.3} 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 {9}, headings {4}, and list marks {5}. Users who include BEL characters in their input text (especially in arguments to macros) will receive mangled output.

3.7 Bullets

A bullet (•) is often obtained on a typewriter terminal by using an "o" overstruck by a "+". For compatibility with *troff*, a bullet string is provided by PWB/MM. Rather than overstriking, use the sequence:

```
\*(BU
```

wherever a bullet is desired. Note that the bullet list (.BL) macros {5.3.3.2} use this string to automatically generate the bullets for the list items.

3.8 Dashes, Minus Signs, and Hyphens

Troff has distinct graphics for a dash, a minus sign, and a hyphen, while *nroff* does not. Those who intend to use *nroff* only may use the minus sign ("-") for all three.

Those who wish mainly to use *troff* should follow the escape conventions of {9}.

Those who want to use both formatters must take care during text preparation. Unfortunately, these characters cannot be represented in a way that is both compatible and convenient. We suggest the following approach:

Dash — Type *(EM for each text dash for both *nroff* and *troff*. This string generates an em dash (—) in *troff* and generates "--" in *nroff*. Note that the dash list (.DL) macros {5.3.3.3} automatically generate the em dashes for the list items.

Hyphen — Type "-" and use as is for both formatters. *Nroff* will print it as is, and *troff* will print a true hyphen.

Minus — Type "\-" for a true minus sign, regardless of formatter. *Nroff* will effectively ignore the "\", while *troff* will print a true minus sign.

3.9 Trademark String

A trademark string *(Tm is now available with PWB/MM. This places the letters "TM" one-half line above the text that it follows.

---

3. Every eight characters in *nroff*; every ½ inch in *troff*.

For example:

```
The PWB/UNIX\*(Tm User's Manual is available from the library.
```

yields:
The PWB/UNIX™ User's Manual is available from the library.

3.10 Use of Formatter Requests

Most formatter requests [9] should *not* be used with PWB/MM because PWB/MM provides the corresponding formatting functions in a much more user-oriented and surprise-free fashion than do the basic formatter requests [3.1]. However, some formatter requests *are* useful with PWB/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 for *troff*. 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 and list styles are covered in [5].

4.1 Paragraphs

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

This macro is used to begin two kinds of paragraphs. In a *left-justified* paragraph, the first line begins at the left margin, while in an *indented* paragraph, it is indented five spaces (see below).

A document possesses a *default paragraph style* obtained by specifying ".P" before each paragraph that does *not* follow a heading [4.2]. The default style is controlled by the register *Pt*. The initial value of *Pt* is 0, which always provides left-justified paragraphs. All paragraphs can be forced to be indented by inserting the following at the beginning of the document:

```
.nr Pt 1
```

All paragraphs will be indented except after headings, lists, and displays if the following:

```
.nr Pt 2
```

is inserted at the beginning of the document.

The amount a paragraph is indented is contained in the register *Pi*, whose default value is 5. To indent paragraphs by, say, 10 spaces, 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 space (½ a vertical space).

☛ *Values that specify indentation must be* unscaled *and are treated as "character positions," i.e., as a number of* ens. *In* troff, *an* en *is the number of points (1 point = 1/72 of an inch) equal to half the current point size. In* nroff, *an* en *is equal to the width of a character.*

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

If .P occurs inside a *list*, the indent (if any) of the paragraph is added to the current list indent {5}.

4.2 Numbered Headings

.H level [heading-text]
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.

☛ *There is no need for a .P macro after a .H (or .HU {4.3}), because the .H macro also performs the function of the .P macro. In fact, if a .P follows a .H, the .P is ignored.* {4.2.2.2}.

4.2.1 Normal Appearance. The normal appearance of headings is as shown in this document. The effect of .H varies according to the *level* argument. First-level headings are *preceded* by two blank lines (one vertical space); all others are *preceded* by one blank line (½ a vertical space).

.H 1 heading-text — gives an underlined (bold) heading *followed* by a single blank line (½ a vertical space). The following text begins on a new line and is indented according to the current paragraph type. Full capital letters should normally be used to make the heading stand out.

.H 2 heading-text — yields an underlined (bold) heading followed by a single blank line (½ a 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 $3 \leqslant n \leqslant 7$, produces an underlined (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 first few .H calls of {4}:

```
.H 1 "PARAGRAPHS AND HEADINGS"
.H 2 "Paragraphs"
.H 2 "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 "Bold, Italic, and Underlined Headings."
.H 5 "Control by Level."
```

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 lines (one vertical space) preceding it, and all others have one blank line (½ a 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 {3.1} occurs after the heading. If the heading level is less than or equal to *Hs*, a blank line (½ a 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* {4.1}. 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 line (½ a vertical space) to appear after the first three heading levels, to have no run-in headings, and to force the text following all headings to be left-justified (regardless of the value of *Pt*), the following should appear at the 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 {4.2.2.2}. *Hc* is 0 initially (no centered headings).

4.2.2.4 Bold, Italic, and Underlined Headings.

4.2.2.4.1 Control by Level. Any heading that is underlined by *nroff* is made bold or italic by *troff*. The string *HF* (heading font) contains seven codes that specify the fonts for heading levels 1-7. The legal codes, their interpretations, and the defaults for *HF* are:

Formatter	*HF Code*			*Default HF*
	1	*2*	*3*	
nroff	no underline	underline	underline	3 3 2 2 2 2 2
troff	roman	italic	bold	3 3 2 2 2 2 2

Thus, all levels are underlined in *nroff*; in *troff*, levels 1 and 2 are bold, levels 3 through 7 are italic. The user may reset HF as desired. Any value omitted from the right end of the list is taken to be 1. For example, the following would result in five underlined (bold) levels and two non-underlined (roman) levels:

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

4.2.2.4.2 Nroff Underlining Style. Nroff can underline in two ways. The normal style (.ul request) is to underline only letters and digits. The continuous style (.cu request) underlines all characters, including spaces. By default, PWB/MM attempts to use the continuous style on any heading that is to be underlined and is short enough to fit on a single line. If a heading is to be underlined, but is too long, it is underlined the normal way (i.e., only letters and digits are underlined).

All underlining of headings can be forced to the normal way by using the -rU1 flag when invoking *nroff* {2.4}.

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

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

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. For example:

```
.ds HP 12 12 11 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. Note that the specified values may also be *relative* point-size changes, e.g.:

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

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.

Omitted or zero values imply that the *default* point size will be used for the corresponding heading level.

☛ *Only the point size of the headings is affected. Specifying a large point size without providing increased vertical spacing (via .HX and/or .HZ) may cause overprinting.*

4.2.2.5 Marking Styles—Numerals and Concatenation.

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

The registers named *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 arguments; each argument is a string indicating the type of marking to be used. Legal values and their meanings are shown below; omitted values are interpreted as 1, while illegal values have no effect.

Value	*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 {13.2.1}.

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 three things:

- specifying in the register *Cl* what level headings are to be saved;
- invoking the .TC macro {10.1} at the end of the document;
- and specifying -rB*n* {2.4} on the command line.

Any heading whose level is less than or equal to the value of the register *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 {7} and footnotes {8}. If this happens, the "Out of temp file space" diagnostic {Appendix E} 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 the 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. This page numbering style can be achieved by specifying the flag -rN3 on the command line {9.9}. 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 printed.

4.6 User Exit Macros •

☛ *This section is intended only for users who are accustomed to writing formatter macros.*

.HX dlevel rlevel heading-text
.HZ dlevel rlevel heading-text

The .HX and .HZ macros are the means by which the user obtains a final level of control over the previously-described heading mechanism. PWB/MM does not define .HX 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. All the default actions occur if these macros are not defined. If the .HX or .HZ (or both) are defined by the user, the user-supplied 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 {4.3}, *dlevel* is equal to the contents of register *Hu*, and *rlevel* is 0. In both cases, *heading-text* is the text of the original invocation.

By the time .H calls .HX, it has already incremented the heading counter of the specified level {4.2.2.5}, produced blank line(s) (vertical space) to precede the heading {4.2.2.1}, and accumulated the

"heading mark", i.e., the string of digits, letters, and periods needed for a numbered heading. When .HX is called, all user-accessible registers and strings can be referenced, as well as the following:

string }0 — If *rlevel* is 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 {4.2.2.2}. A value of 0 means that the heading is run-in. A value of 1 means a break (but no blank line) is to follow the heading. A value of 2 means that a blank line (½ a vertical space) is to follow the heading.

string }2 — If register ;0 is 0, this string contains two unpaddable spaces that will be used to separate the (run-in) *heading* from the following *text*. If register ;0 is 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 lines (halves of vertical space) plus the contents of register ;3 as blank lines (halves of vertical space) plus the number of lines of the heading.

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

Change first-level heading mark from format *n.* to *n.*0:
```
.if \\$1=1 .ds }0 \\n(H1.0\□\□
```
(□ stands for a space)

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
```

If temporary string or macro names are used within .HX, care must be taken in the choice of their names {13.1}.

.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 reset counters for footnotes, figures, tables, etc. Another use might be to change a page header or footer. For example:

```
.de HZ
.if \\$1=1 \{.nr :p 0 \" footnotes
.      nr Fg 0 \" figures
.      nr Tb 0 \" tables
.      nr Ec 0 \" equations
.      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 Hl 4
```

☛ *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, e.g., with terms or phrases to be defined.

5.1 Basic Approach

In order to avoid repetitive typing of arguments to describe the appearance of items in a list, PWB/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 five 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 {5.4, Appendix A, Appendix B}.

5.2 Sample Nested Lists

The input for several lists and the corresponding output are shown below. The .AL and .DL macro calls {5.3.3} contained therein are examples of the *list-initialization* macros. This example will help us to explain the material in the following sections. Input text:

```
.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.
```

Output:

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.

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 that make up the list item.

The .LI macro is used with all lists. It normally causes the output of a single blank line (½ a 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 + xxx
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.

☛ *The* mark *must not contain ordinary (paddable) spaces, because alignment of items will be lost if the right margin is justified* {3.3}.

If the *current mark* (in the *current list*) is a null string, and the first argument of .LI is omitted or null, the resulting effect is that of a *hanging indent*, i.e., the first line of the following text is "outdented," starting at the same place where the *mark* would have started {5.3.3.6}.

5.3.2 List End.

.LE [1]

List End restores the state of the list back to that existing just before the most recent list-initialization macro call. If the optional argument is given, the .LE outputs a blank line (½ a vertical space). This option should generally be used only when the .LE is followed by running text, but not when followed by a macro that produces blank lines of its own, such as .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 {5.4}.

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 *Li* (initially 5)[4] spaces from the indent in force when the .AL

is called, thus leaving room for two digits, a period, and two spaces before the text.

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).

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 {4.2.2.5}.[5] If *type* is omitted or null, then "1" is assumed. If *text-indent* is non-null, it is used as the number of spaces from the current indent to the text, i.e., it is used instead of *Li* for this list only. If *text-indent* is null, then the value of *Li* will be used.

If the third argument is given, a blank line (½ a vertical space) will *not* separate the items in the list. A blank line (½ a vertical space) will occur before the first item, however.

5.3.3.2 Bullet List.

.BL [text-indent] [1]

.BL begins a bullet list, in which each item is marked by a bullet (•) followed by one space. If *text-indent* is non-null, it overrides the default indentation—the amount of paragraph indentation as given in the register *Pi* {4.1}.[6]

If a second argument is specified, no blank lines will separate the items in the list.

5.3.3.3 Dash List.

.DL [text-indent] [1]

.DL is identical to .BL, except that a dash is used instead of a bullet.

5.3.3.4 Marked List.

.ML mark [text-indent] [1]

.ML is much like .BL and .DL, but expects the user to specify an arbitrary mark, which may consist of more than a single character. Text is indented *text-indent* spaces if the second argument is not null; otherwise, the text is indented one more space than the width of *mark*. If the third argument is specified, no blank lines will separate the items in the list.

☛ *The* mark *must not contain ordinary (paddable) spaces, because alignment of items will be lost if the right margin is justified* {3.3}.

5.3.3.5 Reference List.

.RL [text-indent] [1]

A .RL call begins an automatically-numbered list in which the numbers are enclosed by square brackets ([]). *Text-indent* may be supplied, as for .AL. If omitted or null, it is assumed to be 6, a convenient value for lists numbered up to 99. If the second argument is specified, no blank lines will separate the items in the list. The list of references {14} was produced using the .RL macro.

---

4. Values that specify indentation must be *unscaled* and are treated as "character positions," i.e., as the number of *ens*.
5. Note that the "0001" format is *not* permitted.
6. So that, in the default case, the text of bullet and dash lists lines up with the first line of indented paragraphs.

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

```
.tr ~
.VL 20 2
.LI mark~1
Here is a description of mark 1;
"mark 1" of the .LI line contains a tilde translated to an unpaddable space in order
to avoid extra spaces between
"mark" and "1" {3.3}.
.LI second~mark
This is the second mark, also using a tilde translated to an unpaddable space.
.LI third~mark~longer~than~indent:
This item shows the effect of a long mark; one space separates the mark
from the text.
.LI ~
This item effectively has no mark because the
tilde following the .LI is translated into a space.
.LE
```

yields:

mark 1 Here is a description of mark 1; "mark 1" of the .LI line contains a tilde translated to an unpaddable space in order to avoid extra spaces between "mark" and "1" {3.3}.

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

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

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

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

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

yields:

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

☛ *The* mark *must not contain ordinary (paddable) spaces, because alignment of items will be lost if the right margin is justified* {3.3}.

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, if necessary, 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 A}. Its arguments are as follows:

Text-indent gives the number of spaces that the text is to be indented from the current indent. Normally, this value is taken from the register *Li* for automatic lists and from the 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).[7] Within the mark area, the mark is *left-justified* if *pad* is 0. If *pad* is greater than 0, say *n*, then *n* blanks are appended to the mark; the *mark-indent* value 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.

Type and *mark* interact to control the type of marking used. If *type* is 0, simple marking is performed using the mark character(s) found in the *mark* argument. If *type* is greater than 0, automatic numbering or alphabetizing is done, and *mark* is then interpreted as the first item in the sequence to be used for numbering or alphabetizing, i.e., it is chosen from the set (1, A, a, I, i) as in {5.3.3.1}. That is:

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

Each non-zero value of *type* from 1 to 6 selects a different way of displaying the items. The following table shows the output appearance for each value of *type:*

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

where *x* is the generated number or letter.

☛ *The* mark *must not contain ordinary (paddable) spaces, because alignment of items will be lost if the right margin is justified* {3.3}.

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

LB-space, the number of blank lines (½ a vertical space) to be output by .LB itself, defaults to 0 if omitted.

---

7. The *mark-indent* argument is typically 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 line *before* each item in the list; such a list is usually terminated with a ".LE 1" to end the list with a blank line. 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 line 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 lines will result.

Appendix A shows the definitions of the list-initialization macros {5.3.3} in terms of the .LB macro. Appendix B illustrates how the user can build upon those macros to obtain other kinds of lists.

6. MEMORANDUM AND RELEASED PAPER STYLES

One use of PWB/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, case numbers, etc.) is entered in the same way for *both* styles; an argument to one macro indicates which style is being used. The following sections describe the macros used to provide this data. The required order is shown in {6.9}.

If neither the memorandum nor released-paper style is desired, the macros described below should be omitted from the input text. If these macros are omitted, the first page will simply have the page header {9} followed by the body of the document.

6.1 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).[8] The title of the memorandum or paper follows the .TL macro and is processed in fill mode {3.1}. Multiple charging case numbers are entered as "sub-arguments" 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 into several lines.

On output, the title appears after the word "subject" in the memorandum style. In the released-paper style, the title is centered and underlined (bold).

6.2 Author(s)

.AU name [initials] [loc] [dept] [ext] [room] [arg] [arg] [arg]

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). For example:

```
.AU "J. J. Jones" JJJ PY 9876 5432 1Z-234
```

In the "from" portion in the memorandum style, the author's name is followed by location and department number on one line and by room number and extension number on the next. The "x" for the extension is added automatically. The printing of the location, department number, extension

8. The "charging case" is the case number to which time was charged for the development of the project described in the memorandum. The "filing case" is a number under which the memorandum is to be filed.

number, and room number may be suppressed on the first page of a memorandum by setting the register *Au* to 0; the default value for *Au* is 1. Arguments 7 through 9, if present, will follow this "normal" author information, each on a separate line. Certain organizations have their own numbering schemes for memoranda, engineer's notes, etc. These numbers are printed after the author's name. This can be done by providing more than six arguments to the .AU macro, e.g.:

```
.AU "S. P. Lename" SPL IH 9988 7766 5H-444 3322.11AB
```

The name, initials, location, and department are also used in the Signature Block {6.11.1}. 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, "Bell Laboratories" and the location are centered. For the case of authors from different locations, see {6.8}.

6.3 TM Number(s)

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

If the memorandum is a Technical Memorandum, the TM numbers are supplied via the .TM macro. Up to nine numbers may be specified. Example:

```
.TM 7654321 77777777
```

This macro call is ignored in the released-paper and external-letter styles {6.6}.

6.4 Abstract

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

In both the memorandum and released-paper styles, the text of the abstract follows the author information and is preceded by the centered and underlined (italic) word "ABSTRACT."

The .AS (abstract start) and .AE (abstract end) macros bracket the (optional) abstract. The first argument to .AS controls the printing of the abstract. If it is 0 or null, the abstract is printed on the first page of the document, immediately following the author information, and is also saved for the cover sheet. If the first argument is 1, the abstract is saved and printed only on the cover sheet. The margins of the abstract are indented on the left and right by five spaces. The amount of indentation can be changed by specifying the desired indentation as the second argument.[9]

Note that headings {4.2, 4.3} and displays {7} are *not* (as yet) permitted within an abstract.

6.5 Other Keywords

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

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

6.6 Memorandum Types

```
.MT [type] [1]
```

9. Values that specify indentation must be *unscaled* and are treated as "character positions," i.e., as the number of *ens*.

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 sheets. Legal codes for *type* and the corresponding values are:

Code	*Value*
.MT ""	no memorandum type is printed
.MT 0	no memorandum type is printed
.MT	MEMORANDUM FOR FILE
.MT 1	MEMORANDUM FOR FILE
.MT 2	PROGRAMMER'S NOTES
.MT 3	ENGINEER'S NOTES
.MT 4	Released-Paper style
.MT 5	External-Letter style
.MT "*string*"	*string*

If *type* indicates a memorandum style, then *value* will be printed after the last line of author information or after the last line of the abstract, if one appears on the first page. If *type* is longer than one character, then the string, itself, will be printed. For example:

.MT "Technical Note #5"

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

The second argument to .MT is used only if the first argument is 4 (i.e., for the released-paper style) as explained in {6.8}.

In the external-letter style (.MT 5), only the date is printed in the upper right corner of the first page. It is expected that preprinted stationery will be used, providing the author's company logotype and address.

6.7 Date and Format Changes

6.7.1 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.

6.7.2 Alternate First-Page Format. One can specify that the words "subject," "date," and "from" (in the memorandum style) be omitted and that an alternate company name be used:

.AF [company-name]

If an argument is given, it replaces "Bell Laboratories", without affecting the other headings. If the argument is *null*, "Bell Laboratories" is suppressed; in this case, extra blank lines are inserted to allow room for stamping the document with a Bell System logo or a Bell Laboratories stamp. .AF with *no* argument suppresses "Bell Laboratories" and the "Subject/Date/From" *headings*, thus allowing output on preprinted stationery.

The only .AF option appropriate for *troff* is to specify an argument to replace "Bell Laboratories" with another name.

6.8 Released-Paper Style

The released-paper style is obtained by specifying:

.MT 4 [1]

This results in a centered, underlined (bold) title followed by centered names of authors. The location of the last author is used as the location following "Bell Laboratories" (unless .AF {6.7.2} specifies a different company). If the optional second argument to .MT is given, then the name of each author is followed by the respective company name and location. The abstract, if present, follows the author

information.

Information necessary for the memorandum style but not for the released-paper style is ignored.

If the released-paper style is utilized, most BTL location codes[10] are defined as strings that are the addresses of the corresponding BTL locations. These codes are needed only until the .MT macro is invoked. Thus, *following* the .MT macro, the user may re-use these string names. In addition, the macros described in {6.11} and their associated lines of input are ignored when the released-paper style is specified.

Authors from non-BTL locations may include their affiliations in the released-paper style by specifying the appropriate .AF *before* each .AU. For example:

```
.TL
A Learned Treatise
.AF "Getem Inc."
.AU "F. Swatter"
.AF "Bell Laboratories"
.AU "Sam P. Lename" "" CB
.MT 4 1
```

6.9 Order of Invocation of "Beginning" Macros

The macros described in {6.1-6.7}, *if present*, must be given in the following order:

```
.ND new-date
.TL [charging-case] [filing-case]
one or more lines of text
.AF [company-name]
.AU name [initials] [loc] [dept] [ext] [room] [arg] [arg] [arg]
.TM [number] ...
.AS [arg] [indent]
one or more lines of text
.AE
.OK [keyword] ...
.MT [type] [1]
```

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. Once .MT has been invoked, *none* of the above macros can be re-invoked because they are removed from the table of defined macros to save space.

6.10 Example

The input text for this manual begins as follows:

```
.TL
P\s-3WB/MM\s0\(emProgrammer's Workbench Memorandum Macros
.AU "D. W. Smith" DWS PY ...
.AU "J. R. Mashey" JRM  MH ...
.MT 4 1
```

10. The complete list is: AK, CP, CH, CB, DR, HO, IN, IH, MV, MH, PY, RR, RD, WV, and WH.

6.11 Macros for the End of a Memorandum

At the end of a memorandum (but not of a released paper), the signatures of the authors and a list of notations[11] can be requested. The following macros and their input are ignored if the released-paper style is selected.

6.11.1 Signature Block.

.SG [arg] [1]

.SG prints the author name(s) after the last line of text, aligned with the "Date/From" block. Three blank lines are left above each name for the actual signature. If no argument is given, the line of reference data[12] will *not* appear following the last line.

A non-null first argument is treated as the typist's initials, and is appended to the reference data. Supply a null argument to print reference data with neither the typist's initials nor the preceding hyphen.

If there are several authors and if the second argument is given, then the reference data is placed on the same line as the name of the first author, rather than on the line that has the name of the last author.

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

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

6.11.2 "Copy to" and Other Notations.

```
.NS [arg]
zero or more lines of the notation
.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 *arg* and the corresponding notations are:

---

11. See [2], pp. 1.12–16
12. The following information is known as reference data: location code, department number, author's initials, and typist's initials, all separated by hyphens. See [2], page 1.11

Code	*Notations*
.NS ""	Copy to
.NS 0	Copy to
.NS	Copy to
.NS 1	Copy (with att.) to
.NS 2	Copy (without att.) to
.NS 3	Att.
.NS 4	Atts.
.NS 5	Enc.
.NS 6	Encs.
.NS 7	Under Separate Cover
.NS 8	Letter to
.NS 9	Memorandum to
.NS "*string*"	Copy (*string*) to

If *arg* consists of more than one character, it is placed within parentheses between the words "Copy" and "to." For example:

```
.NS "with att. 1 only"
```

will generate "Copy (with att. 1 only) to" as the notation. More than one notation may be specified before the .NE occurs, because a .NS macro terminates the preceding notation, if any. For example:

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

would be formatted as:

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

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

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

6.12 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. -rL90. This has the effect of making the formatter believe that the page is 90 lines long and therefore giving it more room than usual to place the signature or the notations. This will *only* work for a *single-page* letter or memo.

7. DISPLAYS

Displays are blocks of text that are to be kept together—not split across pages. PWB/MM provides two styles of displays:[13] a *static* (.DS) style and a *floating* (.DF) style. In the *static* style, the display appears

in the same relative position in the output text as it does in the input text; this may result in extra white space at the bottom of the page if the display is too 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 and is *not* indented from the existing margin. The user can specify indentation or centering, as well as fill-mode processing.

Displays and footnotes {8} may *never* be nested, in any combination whatsoever. Although lists {5} and paragraphs {4.1} are permitted, no headings (.H or .HU) can occur within displays or footnotes.

7.1 Static Displays

```
.DS [format] [fill]
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 indentation. The *format* argument to .DS is an integer or letter with the following meanings:

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

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

Code	*Meaning*
""	no-fill mode
0 or N	no-fill mode
1 or F	fill mode

Omitted arguments are taken to be zero.

The standard amount of indentation is taken from the register *Si*, which is initially 5. 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 {4.1}. Even though their initial values are the same, these two registers are independent of one another.

The display format value 3 (CB) centers the *entire display* as a block (as opposed to .DS 2 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*(1)/*neqn*(1) "mark" and "lineup" feature to work with centered equations (see section 7.4 below).

By default, a blank line (½ a vertical space) is placed before and after static and floating displays. These blank lines before and after *static* displays can be inhibited by setting the register *Ds* to 0.

---

13. Displays are processed in an environment that is different from that of the body of the text (see the .ev request in [9]).

7.2 Floating Displays

.DF [format] [fill]
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 {7.1}, 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 line (½ a 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 by *nroff* or *troff*, 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 (see below). The *De* register (see below) controls whether or not text will appear on the current page after a floating display has been produced.

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 (when using section-page numbering) or the end of a document is reached, all displays are automatically removed from the queue and output. This will occur before a .CS or .TC 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. Also note that a wide (full page width) display will never fit in the second column of a two-column document.

The registers, their settings, and their effects are as follows:

Values for *De* Register	
Value	Action
0	DEFAULT: No special action occurs.
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: For any other values the action performed is for the value 1.

Values for *Df* Register	
Value	Action
0	Floating displays will not be output until end of section (when section-page numbering) or end of document.

Values for *Df* Register	
Value	Action
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 exactly 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 current page if there is room. Output exactly 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	DEFAULT: Output a new floating display on the current page if there is room. Output at least one, but as many displays as will fit starting at the top of a new page or column.
	NOTE: For any value greater than 5 the action performed is for the value 5.

7.3 Tables

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

The .TS (table start) and .TE (table end) macros make possible the use of the *tbl*(1) processor [11]. They are used to delimit the text to be examined by *tbl*(1) as well as to set proper spacing around the table. The display function and the *tbl*(1) delimiting function are independent of one another, however, so in order to permit one to keep together blocks that contain any mixture of tables, equations, filled and unfilled text, and caption lines the .TS-.TE block should be enclosed within a display (.DS-.DE). Floating tables may be enclosed inside floating displays (.DF-.DE).

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 multi-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*(1), but of the macro definitions provided by PWB/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.
Title lines
.TH
data
.TE
.TS H
global options;
column descriptors.
Title lines
.TH N
data
.TE
```

will cause the table heading to appear at the top of the first table segment, and no heading to appear at the top of the second segment when both appear on the same page. However, the heading will still appear at the top of each page that the table continues onto. This feature is used when a single table must be broken into segments because of table complexity (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 only appear at the beginning of the table and the top of each new page or column that the table continues onto.

7.4 Equations

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

The equation setters *eqn*(1) and *neqn*(1) [6,7] expect to use the .EQ (equation start) and .EN (equation end) macros as delimiters in the same way that *tbl*(1) uses .TS and .TE; however, .EQ and .EN must occur inside a .DS-.DE pair.

The .EQ macro takes an argument that will be used as a label for the equation. The label will appear at the right margin in the "vertical center" of the general equation.

☛ *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/neqn *"defines," .DS and .DE must* not *be used; otherwise extra blank lines will appear in the output.*

7.5 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 .DS-.DE pairs to automatically number and title figures, tables, and equations. They use registers *Fg, Tb, Ec,* and *Ex,* respectively.[14] As an example, the call:

.FG "This is an illustration"

yields:

Figure 1. This is an illustration

.TB replaces "**Figure**" by "**TABLE**"; .EC replaces "**Figure**" by "**Equation**", and .EX replaces "**Figure**" by "**Exhibit**". Output is centered if it can fit on a single line; otherwise, all lines but the first are indented to line up with the first character of the title. The format of the numbers may be changed using the .af request of the formatter.

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; and if *flag* is 2, *override* replaces the number. For example, to produce figures numbered within sections, supply \n(H1 for *override* on each .FG call, and reset *Fg* at the beginning of each section, as shown in {4.6}.

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.

7.6 List of Figures, Tables, Equations, and Exhibits

A List of Figures, List of Tables, List of Exhibits, and List of Equations may be obtained. They will be printed after the Table of Contents is printed if the number registers *Lf*, *Lt*, *Lx*, and *Le* (respectively) are set to 1. *Lf*, *Lt*, and *Lx* are 1 by default; *Le* is 0 by default.

7.7 Blocks of Filled Text

One can obtain blocks of filled text through the use of .DS or .DF. However, to have the block of filled text *centered* within the current line length, the *tbl*(1) program may be used:

14. The user may wish to reset these registers after each first-level heading {4.6}.

```
  .
  .
  .
.DS 0 1
.TS
center;
lw40 .
T{
  .
  .
  .
T}
.TE
.DE
  .
  .
  .
```

The ".DS 0 1" begins a non-indented, filled display. The *tbl*(1) parameters set up a centered table with a column width of 40 ens. The "T{ ... T}" sequence allows filled text to be input as data within a table.

8. FOOTNOTES

There are two macros that delimit the text of footnotes,[15] a string used to automatically number the footnotes, and a macro that specifies the style of the footnote text.

8.1 Automatic Numbering of Footnotes

Footnotes may be automatically numbered by typing the three characters "\•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.

8.2 Delimiting Footnote Text

There are two macros that delimit the text of each footnote:

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

The .FS (footnote start) marks the beginning of the text of the footnote, and the .FE marks its end. The *label* on the .FS, if present, will be used to mark the footnote text. Otherwise, the number retrieved from the string F will be used. Note that automatically-numbered and user-labeled footnotes may be intermixed. If a footnote is labeled (.FS *label*), the text to be footnoted *must* be followed by *label*, rather than by "\•F". 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. Examples:

1. Automatically-numbered footnote:

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

15. Footnotes are processed in an environment that is different from that of the body of the text (see the .ev request in [9]).

2. Labelled footnote:

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

The text of the footnote (enclosed within the .FS-.FE pair) should *immediately* follow the word to be footnoted in the input text, so that "*F" or *label* occurs at the end of a line of input and the next line is the .FS macro call. It is also good practice to append a unpaddable space {3.3} to "*F" or *label* when they follow an end-of-sentence punctuation mark (i.e., period, question mark, exclamation point).

Appendix C illustrates the various available footnote styles as well as numbered and labeled footnotes.

8.3 Format of Footnote Text *

.FD [arg] [1]

Within the footnote text, the user can control the formatting style by specifying text hyphenation, right margin justification, and text indentation, as well as left- or right-justification of the label when text indenting is used. The .FD macro is invoked to select the appropriate style. The first argument is a number from the left column of the following table. The formatting style for each number is given by the remaining four columns. For further explanation of the first two of these columns, see the definitions of the .ad, .hy, .na, and .nh requests in [9].

0	.nh	.ad	text indent	label left justified
1	.hy	.ad	"	"
2	.nh	.na	"	"
3	.hy	.na	"	"
4	.nh	.ad	no text indent	"
5	.hy	.ad	"	"
6	.nh	.na	"	"
7	.hy	.na	"	"
8	.nh	.ad	text indent	label right justified
9	.hy	.ad	"	"
10	.nh	.na	"	"
11	.hy	.na	"	"

If the first argument to .FD is out of range, the effect is as if .FD 0 were specified. If the first argument is omitted or null, the effect is equivalent to .FD 10 in *nroff* and to .FD 0 in *troff*; these are also the respective initial defaults.

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

.FD "" 1

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

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 (over which the user has control by specifying an *even* argument to .FD), hyphenation across pages is inhibited by PWB/MM.

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. In *troff*, footnotes are set in type that

is two points smaller than the point size used in the body of the text.

8.4 Spacing between Footnote Entries

Normally, one blank line (a three-point 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 lines (a six-point vertical space) to occur between footnotes.

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 header: every page, even page only, and odd page only. Thus the page header may have up to two lines of text: the line that occurs at the top of every page and the line for the even- or odd-numbered page. The same is true for the page footer.

This section first describes the default appearance of page headers and page footers, and then the ways of changing them. We use the term *header (not* qualified by *even* or *odd)* to mean 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 {9.2}. There is no default footer and no even/odd default headers or footers, except as specified in {9.9}.

In a memorandum or a released paper, the page header on the first page is automatically suppressed *provided* a break does *not* occur before .MT is called. The macros and text of {6.9} and of {9} as well as .nr and .ds requests do *not* cause a break and are permitted before the .MT macro call.

9.2 Page Header

```
.PH [arg]
```

For this and for the .EH, .OH, .PF, .EF, .OF macros, the argument is of the form:

```
"'left-part'center-part'right-part'"
```

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. See {9.11} for examples.

The .PH macro specifies the header that is to appear at the top of every page. The initial value (as stated in {9.1}) is the default centered page number enclosed by hyphens. See the top of this page for an example of this default header.

If *debug mode* is set using the flag -rD1 on the command line {2.4}, additional information, printed at the top left of each page, is included in the default header. This consists of the SCCS [10] Release and Level of PWB/MM (thus identifying the current version {11.3}), followed by the current line number within the current input file.

9.3 Even-Page Header

```
.EH [arg]
```

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

9.4 Odd-Page Header

```
.OH [arg]
```

This macro is the same as .EH, except that it applies to odd-numbered pages.

9.5 Page Footer

.PF [arg]

The .PF macro specifies the line that is to appear at the bottom of each page. Its initial value is a blank line. If the -rC*n* flag is specified on the command line {2.4}, the type of copy *follows* the footer on a separate line. In particular, if -rC3 (DRAFT) is specified, then, in addition, the footer is initialized to contain the date {6.7.1}, instead of being a blank line.

9.6 Even-Page Footer

.EF [arg]

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

9.7 Odd-Page Footer

.OF [arg]

This macro is the same as .EF, except that it applies to odd-numbered pages.

9.8 Footer on the First Page

By default, the footer is a blank line. If, in the input text, 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 flag is specified on the command line {2.4}.

9.9 Default Header and Footer with "Section-Page" Numbering

Pages can be numbered sequentially within sections {4.5}. To obtain this numbering style, specify -rN3 on the command line. In this case, the default *footer* is a centered "section-page" number, e.g. 3-5, and the default page header is blank.

9.10 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) backslashes. 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 footer with the "section-page" style, one specifies (see {4.2.2.5} for meaning of *H1*):

```
.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.

9.11 Header and Footer Example •

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'"
```

9.12 Generalized Top-of-Page Processing •

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

During header processing, PWB/MM invokes two user-definable macros. One, the .TP macro, is invoked in the environment (see .ev request in [9]) of the header; the other, .PX, 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
.tl \\*()t
.if e 'tl \\*()e
.if o 'tl \\*()o
.sp
..
```

The string }*t* contains the header, the string }*e* contains the even-page header, and the string }*o* contains the odd-page header, as defined by the .PH, .EH, and .OH macros, respectively. To obtain more specialized page titles, the user may redefine the .TP macro to cause any desired header processing [11.5]. Note that formatting done within the .TP macro is processed in an environment different from that of the body.

For example, to obtain a page header that includes three centered lines of data, say, 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. 7, SEP 1977
.sp
..
```

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

9.13 Generalized Bottom-of-Page Processing

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

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

```
.BS
.BE
```

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. The appropriate -rB*n* option {2.4} must *also* be specified on the command line. These macros should normally appear only once at the *end* of the document, after the Signature Block {6.11.1} and Notations {6.11.2} 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 {4.4}. Note that -rB1 or -rB3 {2.4} must also be specified to the formatter on the command line. 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."

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 lines (halves of a vertical space) before them. Both *slevel* and *spacing* default to 1. This means that first-level headings are preceded by one blank line (½ a 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 {4.4}.

The third and fourth arguments control the placement of the page number for each heading. The page numbers can be justified at the right margin with either blanks or dots ("leaders") separating the heading text from the page number, or the page numbers can follow the heading text. For headings whose level is less than or equal to *tlevel* (default 2), 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*, etc.), 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. By defining .TX 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, the following input:

```
.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

⋮

10.2 Cover Sheet

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

The .CS macro generates a cover sheet in either the TM or released-paper style.[16] All of the other information for the cover sheet is obtained from the data given before the .MT macro call {6.9}. If the released-paper style is used, all arguments to .CS are ignored. If a memorandum style is used, the .CS macro generates the "Cover Sheet for Technical Memorandum." The arguments provide the data that appears in the lower left corner of the TM cover sheet [2]: the number of pages of text, the number of other pages, the total number of pages, the number of figures, the number of tables, and the number of references.

11. MISCELLANEOUS FEATURES

11.1 Bold, Italic, and Roman

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

When called without arguments, .B (or .I) changes the font to bold (or italic) in *troff*, and initiates underlining in *nroff*.[17] This condition continues until the occurrence of a .R, when the regular roman font is restored. Thus,

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

yields:

here is some text.

If .B or .I is called with one argument, that argument is printed in the appropriate font (underlined in *nroff*). Then the *previous* font is restored (underlining is turned off in *nroff*). If two arguments are given to a .B or .I, the second argument is then concatenated to the first with no intervening space, but is printed in the previous font (not underlined in *nroff*). For example:

16. But only if -rB2 or -rB3 has been specified on the command line.

17. For ease of explanation, in this section {11.1} *troff* behavior is described first, the convention of {1.2} not withstanding.

```
.I italic
text
.I right -justified
```

produces:

italic text *right*-justified

One can use both bold and italic fonts if one intends to use *troff*, but the *nroff* version of the output does not distinguish between bold and italic. It is probably a good idea to use .I only, unless bold is truly required. Note that font changes in headings are handled separately {4.2.2.4.1}.

Anyone using a terminal that cannot underline might wish to insert:

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

at the beginning of the document to eliminate *all* underlining.

11.2 Justification of Right Margin

```
.SA [arg]
```

The .SA macro is used to set right-margin justification for the main body of text. Two justification flags are used: *current* and *default*. .SA 0 sets both flags to no justification, i.e., it acts like the .na request. .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 a .na or .ad, depending on what the *default* is. Initially, both flags are set for no justification in *nroff* and for justification in *troff*.

In general, the request .na 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.

11.3 SCCS Release Identification

The string *RE* contains the SCCS [10] Release and Level of the current version of PWB/MM. For example, typing:

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

produces:

This is version 15.103 of the macros.

This information is useful in analyzing suspected bugs in PWB/MM. The easiest way to have this number appear in your output is to specify -rD1 {2.4} on the command line, which causes the string *RE* to be output as part of the page header {9.2}.

11.4 Two-Column Output

PWB/MM can print two columns on a page:

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

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. Footnote and display width is controlled by a macro, .WC (Width Control), which takes the following arguments:

N	Normal default mode (−WF, −FF, −WD)
WF	Wide Footnotes always (even in two column mode)
−WF	DEFAULT: turn off WF (footnotes follow column mode, wide in 1C mode, narrow in 2C mode, unless FF is set)
FF	First Footnote; all footnotes have the same width as the *first* footnote encountered for that page
−FF	DEFAULT: turn off FF (footnote style follows the settings of WF or −WF)
WD	Wide Displays always (even in two column mode)
−WD	DEFAULT: Displays follow whichever column mode is in effect when the display is encountered

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. Note that by default all options are turned off.

11.5 Column Headings for Two-Column Output •

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

In two-column output, it is sometimes necessary to have headers over each column, as well as headers over the entire page {9}. This is accomplished by redefining the .TP macro {9.12} 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 16C 31R 34 50C 65R
left→center→right→left→center→right          (where → stands for the tab character)
→first column→→→second column
.fi
.sp 2
..
```

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 65-en overall line length.

11.6 Vertical Spacing

.SP [lines]

There exist several ways of obtaining vertical spacing, 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 typically set at the end of a page header in order to eliminate spacing by a .sp or .bp request that just happens to occur at the top of a page. This mode can be turned

off via the .rs ("restore spacing") request.

The .SP macro is used to avoid the accumulation of vertical space by successive macro calls. Several .SP calls in a row produce *not* the sum of their arguments, but their maximum; i.e., the following produces only 3 blank lines:

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

Many PWB/MM macros utilize .SP for spacing. For example, ".LE 1" {5.3.2} immediately followed by ".P" {4.1} produces only a single blank line (½ a vertical space) between the end of the list and the following paragraph. An omitted argument defaults to one blank line (*one* vertical space). Unscaled fractional amounts are permitted; like .sp, .SP is also inhibited by the .ns request.

11.7 Skipping Pages

.SK [pages]

The .SK macro skips pages, but retains the usual header and footer processing. If *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. .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.

11.8 FORCING AN ODD PAGE

.OP

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

11.9 Setting Point Size and Vertical Spacing

In *troff*, the default point size (obtained from the register *S* {2.4}) is 10, with a vertical spacing of 12 points (i.e., 6 lines per inch). The prevailing point size and vertical spacing may be changed by invoking the .S macro:

.S [arg]

If *arg* is null, the *previous* point size is restored. If *arg* is negative, the point size is decremented by the specified amount. If *arg* is *signed* positive, the point size is incremented by the specified amount, and if *arg* is unsigned, it is used as the new point size; if *arg* is greater than 99, the *default* point size (10) is restored. Vertical spacing is always two points greater than the point size.[18]

12. ERRORS AND DEBUGGING

12.1 Error Terminations

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

- A break occurs.

18. Footnotes {8} are printed in a size two points *smaller* than the point size of the body, with an additional vertical spacing of three points between footnotes.

- To avoid confusion regarding the location of the error, the formatter output buffer (which may contain some text) is printed.
- A short 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 the error messages are explained in Appendix E.)
- Processing terminates, unless the register *D* {2.4} has a positive value. In the latter case, processing continues even though the output is guaranteed to be deranged from that point on.

☛ *The error message is printed by writing it directly to the user's terminal. If an output filter, such as* 300(1), 450(1), *or* hp(1) *is being used to post-process* nroff *output, the message may be garbled by being intermixed with text held in that filter's output buffer.*

☛ *If either* tbl(1) *or* eqn(1)/neqn(1), *or both are being used, and if the* -olist *option of the formatter causes the last page of the document* not *to be printed, a harmless "broken pipe" message results.*

12.2 Disappearance of Output

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

The following command:

```
grep -n "^\.[EDFT][EFNQS]" files ...
```

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

13. EXTENDING AND MODIFYING THE MACROS •

13.1 Naming Conventions

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

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

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

Note that *request, macro,* and *string* names are kept by the formatters in a single internal table, so that there must be no duplication among such names. *Number register* names are kept in a separate table.

13.1.1 Names Used by Formatters.

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

13.1.2 Names Used by PWB/MM.

macros:	AA (most common, accessible to user)
	A (less common, accessible to user)

)x (internal, constant)
>x (internal, dynamic)

strings:
AA (most common, accessible to user)
A (less common, accessible to user)
]x (internal, usually allocated to specific functions throughout)
}x (internal, more dynamic usage)

registers:
Aa (most common, accessible to users)
An (common, accessible to user)
A (accessible, set on command line)
:x (mostly internal, rarely accessible, usually dedicated)
;x (internal, dynamic, temporaries)

13.1.3 Names Used by EQN/NEQN and TBL. The equation preprocessors, *eqn*(1) and *neqn*(1), use registers *and* string names of the form *nn*. The table preprocessor, *tbl*(1), uses names of the form:

a- a+ a| nn #a ## #- #^ ^a T& TW

13.1.4 User-Definable Names. After the above, what is left for user extensions? To avoid problems, we suggest using names that consist either of a single lower-case letter, or of a lower-case letter followed by anything other than a lower-case letter. The following is a sample naming convention:

macros:
aA
Aa

strings:
a
a) (or a], or a}, etc.)

registers
a
aA

13.2 Sample Extensions

13.2.1 Appendix Headings. The following gives a way of generating and numbering appendices:

```
.nr Hu 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 page header changed to "Appendix *a* - *n*") and generates an unnumbered heading of *title*, which, if desired, can be saved for the table of contents. Those who wish Appendix titles to be centered must, in addition, set the register *Hc* to 1 {4.2.2.3}.

13.2.2 Hanging Indent with Tabs. The following example illustrates the use of the hanging-indent feature of variable-item lists {5.3.3.6}. First, a user-defined macro is built to accept four arguments that make up the *mark*. Each argument is to be separated from the previous one by a tab character; 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.[19] The "\t" is

translated by the formatter into a tab character. The "\c" is used to concatenate the line of *text* that follows the macro to the line of text built by the macro. The macro definition and an example of its use are as follows:

```
.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 .hy on \- no
Hyphenate.
Automatic hyphenation is turned on.
.aX .hc\□c none none no                    (□ stands for a space)
Hyphenation indicator character is set to "c" or removed.
During text processing the indicator is suppressed
and will not appear in the output.
Prepending the indicator to a word has the effect
of preventing hyphenation of that word.
.LE
```

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.
.hy	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. Prepending the indicator to a word has the effect of preventing hyphenation of that word.

14. CONCLUSION

The following are the qualities that we have tried to emphasize in PWB/MM, in approximate order of importance:

- *Robustness in the face of error*—A user need not be an *nroff/troff* expert to use these macros. When the input is incorrect, either the macros attempt to make a reasonable interpretation of the error, or a message describing the error is produced. We have tried to minimize the possibility that a user would get cryptic system messages or strange output as a result of simple errors.

---

19. The two-character sequence "\&" is understood by the formatters to be a "zero-width" space, i.e., it causes no output characters to appear.

- *Ease of use for simple documents*—It is not necessary to write complex sequences of commands to produce simple documents. Reasonable default values are provided, where at all possible.
- *Parameterization*—There are many different preferences in the area of document styling. Many parameters are provided so that users can adapt the output to their respective needs over a wide range of styles.
- *Extension by moderately expert users*—We have made a strong effort to use mnemonic naming conventions and consistent techniques in the construction of the macros. Naming conventions are given so that a user can add new macros or redefine existing ones, if necessary.
- *Device independence*—The most common use of PWB/MM is to print documents on hard-copy typewriter terminals, using the *nroff* formatter. The macros can be used conveniently with both 10- and 12-pitch terminals. In addition, output can be scanned with an appropriate CRT terminal. The macros have been constructed to allow compatibility with *troff*, so that output can be produced both on typewriter-like terminals and on a phototypesetter.
- *Minimization of input*—The design of the macros attempts to minimize repetitive typing. For example, if a user wants to have a blank line after all first- or second-level headings, he or she need only set a specific parameter *once* at the beginning of a document, rather than add a blank line after each such heading.
- *Decoupling of input format from output style*—There is but one way to prepare the input text, although the user may obtain a number of output styles by setting a few global flags. For example, the .H macro is used for all numbered headings, yet the actual output style of these headings may be made to vary from document to document or, for that matter, within a single document.

Future releases of PWB/MM will provide additional features that are found to be useful. The authors welcome comments, suggestions, and criticisms of the macros and of this manual.

Acknowledgements. We are indebted to T. A. Dolotta for his continuing guidance during the development of PWB/MM. We also thank our many users who have provided much valuable feedback, both about the macros and about this manual. Many of the features of PWB/MM are patterned after similar features in a number of earlier macro packages, and, in particular, after one implemented by M. E. Lesk. Finally, because PWB/MM often approaches the limits of what is possible with the text formatters, during the implementation of PWB/MM we have generated atypical requirements and encountered unusual problems; we thank J. F. Ossanna for his willingness to add new features to the formatters and to invent ways of having the formatters perform unusual but desired actions.

References

[1] Dolotta, T. A., Haight, R. C., and Piskorik, E. M., eds. *PWB/UNIX User's Manual—Edition 1.0.* Bell Laboratories, May 1977.

[2] Bell Laboratories, Methods and Systems Department. Office Guide. Unpublished Memorandum, Bell Laboratories, April 1972 (as revised).

[3] Kernighan, B. W. UNIX for Beginners. Bell Laboratories, October 1974.

[4] Kernighan, B. W. A Tutorial Introduction to the UNIX Text Editor. Bell Laboratories, October 1974.

[5] Kernighan, B. W. A TROFF Tutorial. Bell Laboratories, August 1976.

[6] Kernighan, B. W., and Cherry, L. L. Typesetting Mathematics—User's Guide (Second Edition). Bell Laboratories, June 1976.

[7] Scrocca, C. New Graphic Symbols for EQN and NEQN. Bell Laboratories, September 1976.

[8] Smith, D. W., and Piskorik, E. M. Typing Documents with PWB/MM. Bell Laboratories, October 1977.

[9] Ossanna, J. F. NROFF/TROFF User's Manual. Bell Laboratories, October 1976.

[10] Bonanni, L. E., and Glasser, A. L. SCCS/PWB User's Manual. Bell Laboratories, November 1977.

[11] Lesk, M. Tbl—A Program to Format Tables. Bell Laboratories, September 1977.

Appendix A: DEFINITIONS OF LIST MACROS •

☛ *This appendix is intended only for users accustomed to writing formatter macros.*

Here are the definitions of the list-initialization macros {5.3.3}:[20]

```
.de AL
.if!@\\$1@@ .if!@\\$1@1@ .if!@\\$1@a@ .if!@\\$1@A@ .if!@\\$1@I@ .if!@\\$1@i@ .)D"AL:bad arg:\\$1
.if \\n(.$<3 \{.ie \w@\\$2@=0 .)L \\n(Lin 0 \\n(Lin-\w@\0\0.@u 1 "\\$1"
.el .LB 0\\$2 0 2 1 "\\$1" \}
.if \\n(.$>2 \{.ie \w@\\$2@=0 .)L \\n(Lin 0 \\n(Lin-\w@\0\0.@u 1 "\\$1" 0 1
.el .LB 0\\$2 0 2 1 "\\$1" 0 1 \}
..
.de BL
.nr ;0 \\n(Pi
.if \\n(.$>0 .if \w@\\$1@>0 .nr ;0 0\\$1
.if \\n(.$<2 .LB \\n(;0 0 1 0 \\•(BU
.if \\n(.$>1 .LB \\n(;0 0 1 0 \\•(BU 0 1
.rr ;0
..
.de DL
.nr ;0 \\n(Pi
.if \\n(.$>0 .if \w@\\$1@>0 .nr ;0 0\\$1
.if \\n(.$<2 .LB \\n(;0 0 1 0 \(em
.if \\n(.$>1 .LB \\n(;0 0 1 0 \(em 0 1
.rr ;0
..
.de ML
.if !\\n(.$ .)D "ML:missing arg"
.nr ;0 \w@\\$1@u/3u/\\n(.su+1u\" get size in n's
.if !\\n(.$-1 .LB \\n(;0 0 1 0 "\\$1"
.if \\n(.$-1 .if !\\n(.$-2 .LB 0\\$2 0 1 0 "\\$1"
.if \\n(.$-2 .if !\w@\\$2@ .LB \\n(;0 0 1 0 "\\$1" 0 1
.if \\n(.$-2 .if \w@\\$2@ .LB 0\\$2 0 1 0 "\\$1" 0 1
..
.de RL
.nr ;0 6
.if \\n(.$>0 .if \w@\\$1@>0 .nr ;0 0\\$1
.if \\n(.$<2 .LB \\n(;0 0 2 4
.if \\n(.$>1 .LB \\n(;0 0 2 4 1 0 1
.rr ;0
..
.de VL
.if !\\n(.$ .)D "VL:missing arg"
.if !\\n(.$-2 .LB 0\\$1 0\\$2 0 0
.if \\n(.$-2 .LB 0\\$1 0\\$2 0 0 \& 0 1
..
```

20. On this page, • represents the BEL character, .)D is an internal PWB/MM macro that prints error messages, and .)L is similar to .LB, except that it expects its arguments to be scaled.

Any of these can be redefined to produce different behavior: e.g., to provide two spaces between the bullet of a bullet item and its text, redefine .BL as follows before invoking it:[21]

```
.de BL
.LB 3 0 2 0 \\•(BU
..
```

21. With this redefinition, .BL cannot have any arguments.

Appendix B: USER-DEFINED LIST STRUCTURES •

☛ *This appendix is intended only for users accustomed to writing formatter macros.*

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

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

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

```
.aL
.LI
first line.
.aL
.LI
second line.
.LE
.LI
third line.
.LE
```

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

For .bL to work, the previous definition of the .aL macro must be changed to obtain the value of *g* from its argument, rather than from *:g*. Invoking .bL without arguments causes it to stay at the current list level. The PWB/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 0". If text is to be resumed at the end of a list, insert the call ".LC 0" to clear out the lists completely. The example below illustrates the relatively small amount of input needed by this approach. The input text:

```
The quick brown fox jumped over the lazy dog's back.
.bL 1
first line.
.bL 2
second line.
.bL 1
third line.
.bL
fourth line.
.LC 0
fifth line.
```

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 FOOTNOTES

The following example illustrates several footnote styles and both labeled and automatically-numbered footnotes. The actual input for the immediately following text and for the footnotes at the bottom of this page is shown on the following page:

With the footnote style set to the *nroff* default, we process a footnote[1] followed by another one.[*****] Using the .FD macro, we changed the footnote style to hyphenate, right margin justification, indent, and left justify the label. Here is a footnote,[2] and another.[†] The footnote style is now set, again via the .FD macro, to no hyphenation, no right margin justification, no indentation, and with the label left-justified. Here comes the final one.[3]

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

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

2. This is the third footnote example (.FD 1). The right margin is justified, the footnote text is indented, the label is *left*-justified in the text-indent space. Although not necessarily illustrated by this example, hyphenation is permitted. The quick brown fox jumped over the lazy dog's back.

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

3. This is the fifth footnote example (.FD 6). The right margin is *not* justified, hyphenation is *not* permitted, the footnote text is *not* indented, and the label is placed at the beginning of the first line. The quick brown fox jumped over the lazy dog's back. Now is the time for all good men to come to the aid of their country.

```
.FD 10
With the footnote style set to the
.I nroff
default, we process a footnote\*F
.FS
This is the first footnote text example (.FD 10).
This is the default style for
.I nroff.
The right margin is
.I not
justified.
Hyphenation is
.I not
permitted.
The text is indented, and the automatically generated label is
.I right -justified
in the text-indent space.
.FE
followed by another one.*****\□                    (□ stands for a space)
.FS *****
This is the second footnote text example (.FD 10).
This is also the default
.I nroff
style but with a long footnote label provided by the user.
.FE
.FD 1
Using the .FD macro, we changed the footnote style to hyphenate, right margin justification,
indent, and left justify the label.
Here is a footnote,\*F
.FS
This is the third footnote example (.FD 1).
The right margin is justified, the footnote text is indented, the label is
.I left -justified
in the text-indent space.
Although not necessarily illustrated by this example, hyphenation is permitted.
The quick brown fox jumped over the lazy dog's back.
.FE
and another.\(dg\□
.FS \(dg
This is the fourth footnote example (.FD 1).
The style is the same as the third footnote.
.FE
.FD 6
The footnote style is now set, again via the .FD macro, to no hyphenation, no right margin justification,
no indentation, and with the label left-justified.
Here comes the final one.\*F\□
.FS
This is the fifth footnote example (.FD 6).
The right margin is
.I not
justified, hyphenation is
.I not
permitted, the footnote text is
.I not
indented, and the label is placed at the beginning of the first line.
The quick brown fox jumped over the lazy dog's back.
Now is the time for all good men to come to the aid of their country.
.FE
```

Appendix D: SAMPLE LETTER

☛ *The* nroff *and* troff *outputs corresponding to the input text below are shown on the following pages.*

```
.ND "May 31, 1979"
.TL 334455
Out-of-Hours Course Description
.AU "D. W. Stevenson" DWS PY 9876 5432 1X-123
.MT 0
.DS
J. M. Jones:
.DE
.P
Please use the following description for the Out-of-Hours course
"Document Preparation on the PWB/UNIX"
.FS "
UNIX is a Trademark of Bell Laboratories.
.FE
time-sharing system":
.P
The course is intended for clerks, typists, and others
who intend to use the PWB/UNIX system
for preparing documentation.
The course will cover such topics as:
.VL 18
.LI Environment:
utilizing a time-sharing computer system;
accessing the system;
using appropriate output terminals.
.LI Files:
how text is stored on the system;
directories;
manipulating files.
.LI "Text editing:"
how to enter text so that subsequent revisions are easier to make;
how to use the editing system to
add, delete, and move lines of text;
how to make corrections.
.LI "Text processing:"
basic concepts;
use of general-purpose formatting packages.
.LI "Other facilities:"
additional capabilities useful to the typist such as the
.I "typo, spell, diff,"
and
.I grep
commands and a desk-calculator package.
.LE
.SG jrm
.NS
S. P. Lename
H. O. Del
M. Hill
.NE
```

Bell Laboratories

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

date: May 31, 1979

from: D. W. Stevenson
PY 9876
1X-123 x5432

J. M. Jones:

Please use the following description for the Out-of-Hours course "Document Preparation on the PWB/UNIX* time-sharing system":

The course is intended for clerks, typists, and others who intend to use the PWB/UNIX system for preparing documentation. The course will cover such topics as:

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

PY-9876-DWS-jrm

D. W. Stevenson

Copy to
S. P. Lename
H. O. Del
M. Hill

* UNIX is a Trademark of Bell Laboratories.

Bell Laboratories

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

date: **May 31, 1979**

from: **D. W. Stevenson**
PY 9876
1X-123 x5432

J. M. Jones:

Please use the following description for the Out-of-Hours course "Document Preparation on the PWB/UNIX* time-sharing system":

The course is intended for clerks, typists, and others who intend to use the PWB/UNIX system for preparing documentation. The course will cover such topics as:

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

PY-9876-DWS-jrm

D. W. Stevenson

Copy to
S. P. Lename
H. O. Del
M. Hill

* UNIX is a Trademark of Bell Laboratories.

Appendix E: ERROR MESSAGES

I. PWB/MM Error Messages

Each PWB/MM error message consists of a standard part followed by a variable part. The standard part is of the form:

ERROR:input line *n*:

The variable part consists of a descriptive message, usually beginning with a macro name. The variable parts are listed below in alphabetical order by macro name, each with a more complete explanation:[22]

Check TL, AU, AS, AE, MT sequence	The proper sequence of macros for the beginning of a memorandum is shown in {6.9}. Something has disturbed this order.
AL:bad arg:value	The argument to the .AL macro is not one of 1, A, a, I, or i. The incorrect argument is shown as *value.*
CS:cover sheet too long	The text of the cover sheet is too long to fit on one page. The abstract should be reduced or the indent of the abstract should be decreased {6.4}.
DS:too many displays	More than 26 floating displays are active at once, i.e., have been accumulated but not yet output.
DS:missing FE	A display starts inside a footnote. The likely cause is the omission (or misspelling) of a .FE to end a previous footnote.
DS:missing DE	.DS or .DF occurs within a display, i.e., a .DE has been omitted or mistyped.
DE:no DS or DF active	.DE has been encountered but there has not been a previous .DS or .DF to match it.
FE:no FS	.FE has been encountered with no previous .FS to match it.
FS:missing FE	A previous .FS was not matched by a closing .FE, i.e., an attempt is being made to begin a footnote inside another one.
FS:missing DE	A footnote starts inside a display, i.e., a .DS or .DF occurs without a matching .DE.
H:bad arg:value	The first argument to .H must be a single digit from 1 to 7, but *value* has been supplied instead.
H:missing FE	A heading macro (.H or .HU) occurs inside a footnote.
H:missing DE	A heading macro (.H or .HU) occurs inside a display.
H:missing arg	.H needs at least 1 argument.
HU:missing arg	.HU needs 1 argument.
LB:missing arg(s)	.LB requires at least 4 arguments.
LB:too many nested lists	Another list was started when there were already 6 active lists.
LE:mismatched	.LE has occurred without a previous .LB or other list-initialization macro {5.3.3}. Although this is not a fatal error,

22. This list is set up by ".LB 37 0 2 0" {5.4}.

	the message is issued because there almost certainly exists some problem in the preceding text.
LI:no lists active	.LI occurs without a preceding list-initialization macro. The latter has probably been omitted, or has been separated from the .LI by an intervening .H or .HU.
ML:missing arg	.ML requires at least 1 argument.
ND:missing arg	.ND requires 1 argument.
SA:bad arg:value	The argument to .SA (if any) must be either 0 or 1. The incorrect argument is shown as *value*.
SG:missing DE	.SG occurs inside a display.
SG:missing FE	.SG occurs inside a footnote.
SG:no authors	.SG occurs without any previous .AU macro(s).
VL:missing arg	.VL requires at least 1 argument.

II. Formatter Error Messages

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

"Cannot do ev" is caused by *(a)* setting a page width that is negative or extremely short, *(b)* setting a page length that is negative or extremely short, *(c)* reprocessing a macro package (e.g. performing a .so to a macro package that was requested from the command line), and *(d)* requesting the −s1 option to *troff* on a document that is longer than ten pages.

"Cannot open *filename*" is issued if one of the files in the list of files to be processed cannot be opened.

"Exception word list full" indicates that too many words have been specified in the hyphenation exception list (via .hw requests).

"Line overflow" means that the output line being generated was too long for the formatter's line buffer. The excess was discarded. See the "Word overflow" message below.

"Non-existent font type" means that a request has been made to mount an unknown font.

"Non-existent macro file" means that the requested macro package does not exist.

"Non-existent terminal type" means that the terminal options refers to an unknown terminal type.

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

"Too many page numbers" is issued when the list of pages specified to the formatter *-o* option is too long.

"Too many string/macro names" is issued when the pool of string and macro names is full. Unneeded strings and macros can be deleted using the .rm request.

"Too many number registers" means that the pool of number register names is full. Unneeded registers can be deleted by using the .rr request.

"Word overflow" means that a word being generated exceeded the formatter's 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*(1)/*neqn*(1).

Appendix F: SUMMARY OF MACROS, STRINGS, AND NUMBER REGISTERS

I. Macros

The following is an alphabetical list of macro names used by PWB/MM. The first line of each item 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 prototype call of the macro.

Macros marked with an asterisk are *not*, in general, invoked directly by the user. Rather, they are "user exits" called from inside header, footer, or other macros.

1C One-column processing {11.4}
.1C

2C Two-column processing {11.4}
.2C

AE Abstract end {6.4}
.AE

AF Alternate format of "Subject/Date/From" block {6.7.2}
.AF [company-name]

AL Automatically-incremented list start {5.3.3.1}
.AL [type] [text-indent] [1]

AS Abstract start {6.4}
.AS [arg] [indent]

AU Author information {6.2}
.AU name [initials] [loc] [dept] [ext] [room] [arg] [arg] [arg]

B Bold (underline in *nroff*) {11.1}
.B [bold-arg] [previous-font-arg]

BE Bottom End {9.13}
.BE

BL Bullet list start {5.3.3.2}
.BL [text-indent] [1]

BS Bottom Start {9.13}
.BS

CS Cover sheet {10.2}
.CS [pages] [other] [total] [figs] [tbls] [refs]

DE Display end {7.1}
.DE

DF Display floating start {7.2}
.DF [format] [fill]

DL Dash list start {5.3.3.3}
.DL [text-indent] [1]

DS Display static start {7.1}
.DS [format] [fill]

EC Equation caption {7.5}
.EC [title] [override] [flag]

EF Even-page footer {9.6}
.EF [arg]

EH Even-page header {9.3}
.EH [arg]

EN End equation display {7.4}
.EN

EQ Equation display start {7.4}
.EQ [label]

EX Exhibit caption {7.5}
.EX [title] [override] [flag]

FD Footnote default format {8.3}
.FD [arg] [1]

FE Footnote end {8.2}
.FE

FG Figure title {7.5}
.FG [title] [override] [flag]

FS Footnote start {8.2}
.FS [label]

H Heading—numbered {4.2}
.H level [heading-text]

HC Hyphenation character {3.4}
.HC [hyphenation-indicator]

HM Heading mark style (Arabic or Roman numerals, or letters) {4.2.2.5}
.HM [arg1] ... [arg7]

HU Heading—unnumbered {4.3}
.HU heading-text

HX * Heading user exit X (before printing heading) {4.6}
.HX dlevel rlevel heading-text

HZ * Heading user exit Z (after printing heading) {4.6}
.HZ dlevel rlevel heading-text

I Italic (underline in *nroff*) {11.1}
.I [italic-arg] [previous-font-arg]

LB List begin {5.4}
.LB text-indent mark-indent pad type [mark] [LI-space] [LB-space]

LC List-status clear {Appendix B}
.LC [list-level]

LE List end {5.3.2}
.LE [1]

LI List item {5.3.1}
.LI [mark] [1]

ML Marked list start {5.3.3.4}
.ML mark [text-indent] [1]

MT Memorandum type {6.6}
.MT [type] [1]

ND New date {6.7.1}
.ND new-date

NE Notation end {6.11.2}
.NE

NS Notation start {6.11.2}
.NS [arg]

OF Odd-page footer {9.7}
.OF [arg]

OH Odd-page header {9.4}
.OH [arg]

OK Other keywords for TM cover sheet {6.5}
.OK [keyword] ...

OP Odd page {11.7}
.OP

P Paragraph {4.1}
.P [type]

PF Page footer {9.5}
.PF [arg]

PH Page header {9.2}
.PH [arg]

PX * Page-header user exit {9.12}
.PX

R Return to regular (roman) font (end underlining in *nroff*) {11.1}
.R

RL Reference list start {5.3.3.5}
.RL [text-indent] [1]

S Set *troff* point size and vertical spacing {11.8}
.S [arg]

SA Set adjustment (right-margin justification) default {11.2}
.SA [arg]

SG Signature line {6.11.1}
.SG [arg] [1]

SK Skip pages {11.7}
.SK [pages]

SP Space—vertically {11.6}
.SP [lines]

TB Table title {7.5}
.TB [title] [override] [flag]

TC Table of contents {10.1}
.TC [slevel] [spacing] [tlevel] [tab] [head1] [head2] [head3] [head4] [head5]

TE Table end {7.3}
.TE

TH Table header {7.3}
.TH [N]

TL Title of memorandum {6.1}
.TL [charging-case] [filing-case]

TM Technical Memorandum number(s) {6.3}
.TM [number] ...

TP * Top-of-page macro {9.12}
.TP

TS Table start {7.3}
.TS [H]

TX * Table-of-contents user exit {10.1}
.TX

VL Variable-item list start {5.3.3.6}
.VL text-indent [mark-indent] [1]

WC Width Control {11.4}
.WC [format]

II. Strings

The following is an alphabetical list of string names used by PWB/MM, giving for each a brief description, section reference, and initial (default) value(s). See {1.4} for notes on setting and referencing strings.

BU Bullet {3.7}
nroff: ⊕
troff: •

F Footnote numberer {8.1}
nroff: \u\\n+ (:p\d
troff: \v'-.4m'\s-3\\n+ (:p\s0\v'.4m'

DT Date (current date, unless overridden) {6.7.1}
Month day, year (e.g., January 22, 1980)

EM Em dash string, produces an em dash for both *nroff* and *troff* {3.8}.

HF Heading font list, up to seven codes for heading levels 1 through 7 {4.2.2.4.1}
3 3 2 2 2 2 2 (all underlined in *nroff*, and B B I I I I I in *troff*)

HP Heading point size list, up to seven codes for heading levels 1 through 7 {4.2.2}

RE SCCS Release and Level of PWB/MM {11.3}
Release.Level (e.g., 15.103)

Tm Trademark string places the letters "TM" one half-line above the text that it follows.

Note that if the released-paper style is used, then, in addition to the above strings, certain BTL location codes are defined as strings; these location strings are needed only until the .MT macro is called {6.8}.

III. Number Registers

This section provides an alphabetical list of register names, giving for each a brief description, section reference, initial (default) value, and the legal range of values (where [m:n] means values from m to n inclusive).

Any register having a single-character name can be set from the command line. An asterisk attached to a register name indicates that that register can be set *only* from the command line or *before* the PWB/MM

macro definitions are read by the formatter {2.4, 2.5}. See {1.4} for notes on setting and referencing registers.

A * Has the effect of invoking the .AF macro without an argument {2.4}
0, [0:1]

Au Inhibits printing of author's location, department, room, and extension in the "from" portion of a memorandum {6.2}
1, [0:1]

B * Defines table-of-contents and/or cover-sheet macros {2.4}
0, [0:3]

C * Copy type (Original, DRAFT, etc.) {2.4}
0 (Original), [0:3]

Cl Contents level (i.e., level of headings saved for table of contents) {4.4}
2, [0:7]

D * Debug flag {2.4}
0, [0:1]

De Display eject register for floating dislays {7.2}
0, [0:1]

Df Display format register for floating displays {7.2}
5, [0:5]

Ds Static display pre- and post-space {7.1}
1, [0:1]

Ec Equation counter, used by .EC macro {7.5}
0, [0:?], incremented by 1 for each .EC call.

Ej Page-ejection flag for headings {4.2.2.1}
0 (no eject), [0:7]

Fg Figure counter, used by .FG macro {7.5}
0, [0:?], incremented by 1 for each .FG call.

Fs Footnote space (i.e., spacing between footnotes) {8.4}
1, [0:?]

H1-H7 Heading counters for levels 1-7 {4.2.2.5}
0, [0:?], incremented by .H of corresponding level or .HU if at level given by register *Hu*. H2-H7 are reset to 0 by any heading at a lower-numbered level.

Hb Heading break level (after .H and .HU) {4.2.2.2}
2, [0:7]

Hc Heading centering level for .H and .HU {4.2.2.3}
0 (no centered headings), [0:7]

Hi Heading temporary indent (after .H and .HU) {4.2.2.2}
1 (indent as paragraph), [0:2]

Hs Heading space level (after .H and .HU) {4.2.2.2}
2 (space only after .H 1 and .H 2), [0:7]

Ht Heading type (for .H: single or concatenated numbers) {4.2.2.5}
0 (concatenated numbers: 1.1.1, etc.), [0:1]

Hu Heading level for unnumbered heading (.HU) {4.3}
2 (.HU at the same level as .H 2), [0:7]

Hy Hyphenation control for body of document {3.4}
1 (automatic hyphenation on), [0:1]

L * Length of page {2.4}
66, [20:?] (11i, [2i:?] in *troff*)[23]

Li List indent {5.3.3.1}
5, [0:?]

Ls List spacing between items by level {5.3.3.1}
5, [0:5]

N * Numbering style {2.4}
0, [0:3]

O * Offset of page {2.4}
0, [0:?] (0.5i, [0i:?] in *troff*)[23]

P Page number, managed by PWB/MM {2.4}
0, [0:?]

Pi Paragraph indent {4.1}
5, [0:?]

Pt Paragraph type {4.1}
2 (paragraphs indented except after headings, lists, and displays), [0:2]

S * *Troff* default point size {2.4}
10, [6:36]

Si Standard indent for displays {7.1}
5, [0:?]

T * Type of *nroff* output device {2.4}
0, [0:2]

Tb Table counter {7.5}
0, [0:?], incremented by 1 for each .TB call.

U * Underlining style (*nroff*) for .H and .HU {2.4}
0 (continuous underline when possible), [0:1]

W * Width of page (line and title length) {2.4}
65, [10:1365] (6.5i, [2i:7.54i] in *troff*)[23]

January 1980

23. For *nroff*, these values are *unscaled* numbers representing lines or character positions; for *troff*, these values must be *scaled*.