PDL/81™

Document Language Reference Guide

(Version 2.0)

DRAFT

Caine, Farber & Gordon, Inc. Warren Point International Ltd.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure of the software described herein is governed by the terms of a license agreement or, in the absence of an agreement, is subject to restrictions stated in subparagraph (c)(1) of the Commercial Computer Software – Restricted Rights clause at FAR 52.227-19 or subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, as applicable. [Caine, Farber & Gordon, Inc.; 1010 East Union St.; Pasadena, CA 91106]

Comments or questions relating to this manual or to the subject software are welcomed and should be addressed to:

In North America:

Caine, Farber & Gordon, Inc.
1010 East Union Street
Pasadena, CA 91106
USA

Tel: (800) 424-3070 or
(818) 449-3070

Fax: (818) 440-1742

In the Rest of the World:

Warren Point International Ltd.
Babbage Road
Stevenage, Herts SG1 2EQ
England

Tel: 0438 316311

Fax: 0227 86521

Form Number: 9102-3

1 August 1988
1 December 1991

Copyright © 1981, 1985, 1988, 1991 by Caine, Farber & Gordon, Inc. All Rights Reserved.

PDL/74, PDL/81, PDL/91, and the PDL prefix are trademarks of Caine, Farber & Gordon, Inc. UNIX is a registered trademark of UNIX System Laboratories. PostScript is a registered trademark of Adobe Systems Incorporated. Ada is a registered trademark of the U. S. Governmenment (Ada Joint Program Office). VAX, VMS, and ULTRIX are trademarks of Digital Equipmeent Corporation. MS and XENIX are trademarks of Microsoft Corporation.

Contents

Appendices

Figures

1. Introduction

PDL/81 is a software tool intended as an aid to designing and documenting a program or system of programs. The tool consists of a processor and a style library which is used to tailor the processor to the particular requirements of the document being produced. As distributed, the style library includes definitions for such document styles as:

- program designs;
- manuals and reports;
- memoranda; and
- business letters.

This manual describes the particular library components which relate to formatting documents other than program designs. In particular, manuals, letters, memoranda, and general text formatting are discussed. Other manuals (see Section 1.3) describe the other library components and the methods for modifying the data base.

1.1 Features and Capabilities of PDL/81

PDL/81 is a tool which integrates the capabilities commonly associated with a program design language processor and those of a text processing system.

This integration is accomplished by providing an extensive set of primitive formatting operations and a definitional language which allows a format designer to compose abstract constructs from these primitive operations. As an example, a document style for program designs might contain such concepts as “data segment” and “flow segment” while a style for manuals might contain such concepts as “chapter”, “enumerated list”, and “paragraph”.

The end user of PDL/81 uses these abstract concepts without any need to understand the underlying implementation or format design methods. However, the underlying format design language allows the local project manager to tailor the document styles and formats to the requirements of a particular project.

The document language described in this manual provides a number of formatting features including:

- automatic line filling and justification
- running page heads and feet with optional formatting for even and odd pages
- automatically formatted title page and title page reverse for manuals
- seven levels of chapter and section headings with user control over placement
- automatically generated table of contents with user control over the levels of headings which will appear
- automatic formatting of figures and tables with optional list of figures and list of tables
- formatting of several different kinds of lists with automatic generation of item numbers in enumerated lists
- automatic paragraph indentation
- automatically formatted displays
- collection and formatting of a document index from user-provided index terms
- automatic formatting of memoranda header blocks
- optional security banners with sheet count control on manuals and other documents
- automatic portion marking of classified documents

1.2 Document Styles and the PDL/81 Style Library

The document styles which are available at an installation reside in the PDL/81 style library. The form of the library depends on the particular host operating system. The particular style to be used in a PDL/81 run is specified as an option when PDL/81 is invoked.

1.3 Related Publications

Other publications relating to the use of PDL/81 are:

- *PDL/81 Introduction and Invocation Guide* – a guide to invoking PDL/81 under various operating environments
- *PDL/81 Design Language Reference Guide* – a guide to using PDL/81 for producing software design documents
- *PDL/81 Ada Design Language Reference Guide* – a guide to using PDL/81 for Ada program design
- *PDL/81 Format Designers Guide* – A guide to developing new types of PDL/81 design and document styles
- *PDL/81 Installation Guide* – a guide to installing PDL/81 under the various supported operating systems.

1.4 A Note to the Reader

The document styles described in this manual are those which we have found to be generally useful. For example, output is intended to be printed in a 12-pitch type font with one-inch left and right margins; title pages and memoranda headings are intended to be printed on our standard forms; and manuals are intended to be printed on both sides of the paper. Thus, the detailed formats meet our requirements and suit our tastes – but you may want something different. If that is the case, feel free to change the data base files. Simple modifications can generally be accomplished after an examination of those files and after reading the *PDL/81 Installation and Tailoring Guide*. Extensive modifications, and the development of entirely new document styles will require reference to the *PDL/81 Format Designers Guide*.

The best way to learn the use of the PDL/81 document language is to construct small examples and run them. Also, the listing of the source of the root file and of Chapter 2 of this manual are presented in Appendix F as they would be input to PDL/81 to process this manual, itself.

From time-to-time, various default values and settings are mentioned in this manual. These are the defaults contained in the distributed style and device definitions. They may be changed when PDL/81 is installed.

2. General Information

This chapter discusses aspects of the various PDL/81 document language styles which are of general interest. It includes information on overall operation of the processor, general input considerations, and the syntax of document language commands and functions.

2.1 Overall Operation

PDL/81 processes a document in one pass through the source. If a table of contents is requested, it will normally be printed at the *end* of the document and can be moved to its proper place for publication. At the expense of additional processing time, however, the table of contents may be printed in-line under control of the "IToc" number register as described in Section 4.4.

During operation, PDL/81 maintains a dictionary of *tags* which are used in referring to some point in a document from some other point (see Section 3.15). At the end of a run, the dictionary is written to an auxiliary file which will be read back in the next time the document is processed.

2.2 Input Format

Input to PDL/81 consists of a sequence of source lines. Each line is terminated by a newline character. The only ASCII control codes allowed are "tab" and "newline". This section describes the interpretation of various special characters and character sequences within source lines.

2.2.1 Tab Expansion on Input

ASCII tab characters are allowed on input lines. Each tab will be replaced by enough blanks to position the immediately following character to the next input tab stop. Input tab stops are set at columns 1, 9, 17, ...

2.2.2 Continuation of Input Lines

The sequence "\<newline>" results in deletion of both characters, thus causing the following line to be considered part of the current line. The character "\" is known as the *escape character* and has additional uses as described throughout this manual.

2.2.3 Special Characters

The escape character may be used to input two special characters – the *bullet* and the *unpaddable space*.

2.2.3.1 The Bullet Character

The special sequence "*" will be replaced by the so-called *bullet* character (bullet) in the output. The actual character printed depends on the particular output device.

2.2.3.2 The Unpaddable Space

The escape character followed by a space is known as the *unpaddable space*. It will be replaced by a single space in the output but will not be subject to size expansion during justification and will not act as a word break.

2.3 Command Lines

If the first character of a line is a period ("."), the line is known as a *command line*. Command lines contain *commands* which direct various types of processing or provide information to PDL/81.

When a command line is encountered, white space (blanks and tabs) following the initial period is skipped. If a newline is encountered, the command line is ignored. If an asterisk ("*") is encountered, the line is considered to be a *comment command*, the rest of the line is skipped, and the whole line is ignored.

If anything else is encountered, it is assumed to start a *command name* which extends to the first blank, tab, or newline. After skipping any white space, the remainder of the line, if any, is taken as one or more *arguments* of the command. Arguments are separated from each other by semicolons (";"). Thus, the general form of a command line is

```
.name [argument[;argument]...]
```

where the brackets indicate optional material.

For example,

```
.Space   5
```

is a command line with a command name of "Space" and an argument of "5".

The case (upper, lower, mixed) of a command name is immaterial. Thus, "Space", "SPACE", "space", or even "sPAce" all represent the same command name.

If a command argument includes a semicolon, the semicolon must be protected by an escape character so that it will not be taken as an argument separator. For example, the command

```
.PTitle arg with ; in it
```

should be entered as

```
.PTitle arg with \; in it
```

if the single argument is to be “arg with ; in it”.

If an input line beginning with a period is not to be taken as a command line, the period must be protected with an escape character. For example, the input line

```
. line with a period as first character
```

should be entered as

```
\. line with a period as first character
```

2.4 Text Functions

Text functions are used to insert special information into a document or to perform some kind of textual modification. The general form of a text function invocation is

```
#{name[;argument[;argument]...]}
```

where the brackets indicate optional information. The case (upper, lower, mixed) of a function name is immaterial. If an argument to a text function contains any of

```
#{   {   }   ;
```

each must be preceded by an escape character. Thus, to invoke function “func” with one argument being “#{” and another being “;;”,

```
#{func;\#\{;\;\;}
```

may be used.

The entire text of a function invocation must be contained on a single (possibly continued) source line. Within a function call, leading white space on a continuation line is ignored.

2.4.1 Nesting of Function Calls

Calls on text functions may be nested. For example, consider the “us” function which underscores its argument and the “cap” function which changes each lowercase letter in its argument to upper case. The sequence

```
#{us;#{cap;this is a test}}
```

or

```
#{cap;#{us;this is a test}}
```

would result in

THIS IS A TEST

2.5 Including Alternate Source Files

At any point in a source file, input may be switched to another source file by the command

```
.Include   file
```

where *file* is the name of the file to be included. Files included with an "Include" command may contain "Include" commands.

2.6 Number Registers

A *number register* is a numeric-valued variable used to provide information to PDL/81. The various number registers used in the document language are discussed throughout this manual.

A number register may be assigned or reassigned a value by the command

```
.Set   name;value
```

where *name* is the name of the number register and *value* is a number giving the value.

For example, the "show" number register controls whether or not processing progress will be displayed on the user's terminal. Within a document, it may be set by the "Set" command as in

```
.Set   show;0
```

which would suppress the display.

The command

```
.Incr   name;value
```

will increment the named number register by the given value, and the command

```
.Decr   name;value
```

will decrement the named number register by the given value.

2.6.1 Global Control Number Registers

Several number registers can be used to control various global aspects of a document as described in this section.

2.6.1.1 Page Offset

The global page offset is contained in the ".po" number register. The value of this register is the number of character positions by which each output line is to be indented, thus shifting the entire document to the right on the output page. The default value of the ".po" number register depends on the selected device. A new value may be assigned by

```
.Set    .po;value
```

2.6.1.2 Use of Output Formfeed Characters

The ".NoFF" number register is a switch which specifies whether or not formfeed control characters may be inserted in the output by PDL/81. If the value of the number register is zero, PDL/81 may insert formfeeds. If the value of the number register is non-zero, PDL/81 will not insert formfeeds. The default value of the number register depends on the selected device. It may be changed by the command

```
.Set    .NoFF;value
```

2.6.1.3 Use of Output Tab Characters

The ".NoTab" number register is a switch which specifies whether or not horizontal tab control characters may be inserted in the output by PDL/81. If the value of the number register is zero, PDL/81 may insert tabs. If the value of the number register is non-zero, PDL/81 will not insert tabs. The default value of the number register depends on the selected device. It may be changed by the command

```
.Set    .NoTab;value
```

2.6.1.4 Use of Output Backspace Characters

The ".NoBs" number register is a switch which specifies whether or not backspace characters may be inserted in the output by PDL/81. If the value of the number register is zero, PDL/81 may insert backspace characters. If the value of the number register is non-zero, PDL/81 will not insert backspace characters and operations such as overstriking and underlining will be performed by other means. The default value of the number register depends on the selected device. It may be changed by the command

```
.Set    .NoBs;value
```

2.6.1.5 Controlling Progress Reports

The “Show” number register is a switch which controls whether or not a report of processing progress is to be displayed on the standard error file. If the value of the number register is zero, a report is not displayed; otherwise, it is displayed. The default value is “1”. It may be changed by the command

```
.Set   Show;value
```

3. General Formatting Operations

This chapter describes a number of formatting operations which are useful for all of the document styles. Included are such topics as dates, underscoring, paragraphing, vertical spacing, headings, lists, and displays.

3.1 Dates

The current date may be accessed within a PDL/81 document and a date may be supplied to use in place of the current date.

3.1.1 Accessing the Date

The date at which the current run of PDL/81 started is available in two forms. The text function

```
#{date}
```

returns the date as 20 September 1991, and the text function

```
#{date3}
```

returns the date as 20 Sep 91. Thus, the line which produces

```
Today is 20 September 1991 (20 Sep 91)
```

can be entered as

```
Today is #{date} (#{date3})
```

3.1.2 Setting the Date

Normally, PDL/81 will supply the current date as the value for the “date” and “date3” text functions. Any desired date in the 20th century may be supplied, instead, by the command

```
.Date   year;month;day
```

If used, this command should occur before any text or any command which results in output. For example, the command

```
.Date   85;12;25
```

will result in the “date” function yielding “25 December 1985” and the “date3” function yielding “25 Dec 85”.

3.2 Times

The time at which the current run of PDL/81 started is available in two forms. The text function

```
#{time}
```

returns the time as 10:15:32 and the text function

```
#{time4}
```

returns the time as 10:15. Thus, the line which produces

```
The time is 10:15:32 (10:15)
```

can be entered as

```
The time is #{time} (#{time4})
```

3.3 Referencing the Page Number

It is sometimes necessary to obtain the current page number – particularly when defining specialized running page heads or feet (see Section 3.9). This can be done by the text function

```
#{page}
```

which returns the current page number as its value.

3.4 Fonts

Words and phrases may be printed in one of several fonts. The functions which perform this cause a switch from the current font to the selected font. For this reason, nesting of the font selection functions will not usually have the expected effect.

3.4.1 Underscoring

The text function

```
#{us;text}
```

causes each non-blank character in *text* to be underscored and the text function

```
#{uc;text}
```

causes each character in *text* to be underscored. For example,

```
this #{us;is under}scored and #{uc;so is this}
```

will print as

this is underscored and so is this

Blanks which are underscored with the “uc” text function do not act as word breaks. Thus, the entire argument of a call on “uc” will be treated as a single word for purposes of layout and justification.

3.4.2 Bold Face Output

If the selected device can support bold face output, the following text may be used to print a word or phrase in bold face. If the device does not support bold face, these functions may still be used, but the output will not be in bold face.

The functions are

```
#{bf;text}
#{bfu;text}
#{bfuc;text}
```

which result in:

bf	print *text* in bold face
bfu	print *text* in bold face with non-blank characters underscored
bfuc	print *text* in bold face with all characters underscored

3.5 Capitalizing

The text function

```
#{cap;text}
```

causes each lower-case letter in *text* to be promoted to upper case. Thus,

```
this is UPPER CASE
```

would result from

```
this is #{cap;upper case}
```

3.6 Formatting Modes

The normal formatting mode of the PDL/81 document language is *filled* and *justified*. In filled mode, input lines are considered to be running text. Words are collected, regardless of input line boundaries, and are put into the output line until a word does not fit. The output line is then printed and a new output line is started. This action is known as "breaking" the line. If justification is in effect, the right end of the broken line will be justified to the right margin of the page by expanding white space in the line.

Since PDL/81 handles all formatting automatically in filled mode, management and future editing of the source will be much easier if each sentence starts on a new input line (even though this is not required). If an input line ends with ".", "?", or "!", it is taken to be the end of a sentence and an extra space will be automatically provided in the output.

In unfilled mode, each input line will start a new output line. If the input line does not fit on one output line, it will be broken at a word boundary and continued to following output lines.

Unfilled mode is selected by the command

```
.NoFill
```

and filled mode may be resumed by the command

```
.Fill
```

Filled mode is always established following any heading command (Section 3.10).

Justification is turned off by the command

```
.NoJustify
```

and is turned on by the command

```
.Justify
```

A line break in filled mode may be forced by the command

```
.Br
```

Many commands, and blank lines, also force a line break.

3.7 Paragraphs

In filled mode, the start of a new paragraph is indicated by a blank (empty) input line. This will cause a one-line interparagraph vertical space and the first line of the paragraph will be indented 5 spaces. The indentation amount is kept in the "Pp_Indent" number register and can be modified with the "Set" command as described in Section 2.6.

3.8 Vertical Space

The command

```
.Sp   n
```

will cause *n* lines of vertical white space to be inserted in the document. If *n* is absent, a value of "1" will be used. Spacing will not be performed at the top of a page and spacing will stop if the bottom of a page is reached.

Spacing requested by successive "Sp" commands is not cumulative but maximal. Thus, the sequence

```
.Sp
.Sp 5
.Sp 2
```

will cause only five lines of vertical white space to be inserted.

The command

```
.Eject   [n]
```

will cause a new output page to be started. If *n* is given, it specifies the number of blank pages to be inserted at this point in the output. These blank pages will, however, have the currently defined page headings and footings. If the output is currently positioned at the top of a page, an "Eject" with a missing or zero-valued *n* will not cause an extra eject. Thus,

```
.Eject   1
```

will leave a blank page while

```
.Eject
.Eject
```

will not.

The command

```
.FloatPage   [n]
```

will cause *n* blank pages (one blank page if *n* is absent) to be inserted in the output the next time the top of an output page is reached. These blank pages will, however, have the currently defined page headings and footings. FloatPage requests are cummulative.

The command

```
.Need   n
```

will cause a new output page to be started if fewer than *n* lines remain on the current page. Thus,

```
.Need   8
```

will do nothing if at least eight lines remain on the current page; otherwise, it will cause a page eject.

3.9 Running Page Heads and Feet

Each document style described in this manual establishes default formats for running page heads and feet which are consistent with the type of document being formatted. These defaults can be changed with the commands described in this section. In these commands, the arguments are:

left	text to be printed flush left
center	text to be printed centered
right	text to be printed flush right

Page headings are specified by

```
.Eh   left;center;right
```

which establishes heads for even numbered pages, by

```
.Oh   left;center;right
```

which establishes heads for odd numbered pages, and by

```
.Ph   left;center;right
```

which establishes heads for both even and odd numbered pages.

Page feet are specified by

```
.Ef   left;center;right
```

which establishes feet for even numbered pages, by

```
.Of   left;center;right
```

which establishes feet for odd numbered pages, and by

```
.Pf   left;center;right
```

which establishes feet for both even and odd numbered pages.

For example, the current date could be printed in the upper left corner of even numbered pages and in the upper right corner of odd numbered pages by the commands

```
.EH #{date3}
.OH ;;#{date3}
```

and the current page number could be printed centered and enclosed in dashes at the bottom of each page by the command

```
.PF ;- #{page} -
```

Heads and feet are normally printed in the base font. This may be changed, how-

ever, as described in Appendix E.

3.10 Numbered Headings

The PDL/81 document language supports seven levels of numbered headings. A numbered heading is set by the command

```
.Hn   text
```

where *n* is one of the digits 1 through 7 and *text* is the text to be used for the heading. Level 1 headings are considered the most important while level 7 headings are considered the least important. Thus, level 7 is subordinate to level 6, level 6 is subordinate to level 5, level 5 is subordinate to level 4, etc.

The Filled mode is always established following one of these commands.

Each heading level has a counter associated with it and the counter is automatically incremented each time a heading command for that level is encountered. At the same time, the counters for all subordinate levels are reset.

The way in which a given heading level is formatted depends upon the particular document style being used. For example, the formatting for the *manual* document style (see Chapter 4) is:

1	Capitalized, underscored, centered, at the top of a new page
2	Capitalized, underscored, flush left, alone on a line
3	Underscored, flush left, alone on a line
4 - 7	Underscored, flush left, followed by a period, run-in

Appropriate vertical spacing is automatically performed before and after each heading. As an example, the current section heading could be set in PDL/81 by

```
.H2 Numbered Headings
```

3.10.1 Changing Heading Defaults

The heading formatting defaults for the various document styles can be changed by manipulating certain number registers with the “Set” command (see Section 2.6).

The number register “EjLevel” specifies the numerically highest heading level which will be centered at the top of a new page. For example, it is set to “1” to cause level 1 headings to be so treated and is set to “0” to cause no headings to be so treated.

The number register “CapLevel” specifies the numerically highest heading level which will be automatically capitalized. For example, it is set to “2” to cause level 1 and level 2 headings to be capitalized and is set to “0” to cause no headings to be capitalized.

The number register “RiLevel” specifies the numerically lowest level heading which will be run-in. For example, it is set to “4” to cause levels 4 through 7 to be run-in.

The number register “OddChap” is a switch which is used only with headings which, under control of the “EjLevel” number register, are placed at the top of a new page. If “OddChap” is non-zero, the new page will always be an odd-numbered page. If “OddChap” is “1” and an even-numbered page must be skipped, the page number will just be incremented – a blank, even-numbered page will not be printed. If “OddChap” is “2” and an even-numbered page must be skipped, a blank page with the current running page heads and feet will be printed. If “OddChap” is “3” and an even-numbered page must be skipped, an entirely blank page will be printed.

The font in which a given level of heading will be printed may be changed as described in Appendix E.

3.11 Unnumbered Headings

The PDL/81 document language supports three styles of unnumbered headings which are completely independent of the numbered heading mechanism described in Section 3.10.

The command

```
.MajorHeading   text
```

will set *text* at the top of a new page, capitalized, underscored, and centered.

The command

```
.Heading   text
```

will set *text* flush left, capitalized, and underscored.

The command

```
.SubHeading   text
```

will set *text* flush left and underscored.

Appropriate vertical spacing is performed before and after each of the unnumbered headings.

3.12 Notes, Cautions, and Warnings

The command

```
.Note [text]
```

will format a centered and boxed heading with *text* in it (or the word “NOTE”, if

text is absent), and will then move the left and right margins inward by the value of the “Nt_Indent” number register. To return to the normal mode, the command

```
.ENote
```

is used. Appropriate vertical spacing is performed before and after the heading and following the “.ENote” command. As an example, consider

CAUTION

> Notes such as this are useful in order to call attention to some particularly important point in a document. It is important, however, that notes not be used too often. If they are, they will soon lose their impact and the reader will tend to simply skip over them.

Such a note is started by

```
.Note CAUTION
```

and ended by

```
.ENote
```

For convenience, the command

```
.Caution
```

is equivalent to

```
.Note   CAUTION
```

and the command

```
.Warning
```

is equivalent to

```
.Note   WARNING
```

3.13 Lists

The PDL/81 document language provides specific generators for the most common kinds of lists as well as three very general commands for other list types. The three general types of lists are:

itemized	each list entry is prefixed by some mark (e.g., a dash or a bullet)
enumerated	each list entry is prefixed by an automatically generated number (which may be displayed in one of several formats)
verb	each list entry is prefixed by an arbitrary word or phrase (this list is an example of a verb list)

The same general structure is used for generating each kind of list as shown in the enumerated list:

1. A *list start* command specifying the kind of list
2. One or more *list items*
3. A *list end* command to mark the end of the list

Lists may be nested.

3.13.1 Itemized Lists

The two predefined types of itemized lists are

- Bullet list, where each item is prefixed with a bullet (this is a bullet list)
- Dashed list, where each item is prefixed with a dash

A generalized itemized list mechanism is provided so that any desired prefix string may be used.

3.13.1.1 Bullet Lists

A bullet list is introduced by the command

```
.BL   [indent]
```

where *indent* is an optional number specifying the amount to indent the text portion of each list item. If *indent* is absent, a default indent amount of 8 will be used. This default indent is kept in the “Li_Indent” number register which may be changed to a new value with the “Set” command (see Section 2.6).

3.13.1.2 Dashed Lists

The dashed list prefixes each list item with a dash (“-”). It is introduced by the command

```
.DL   [indent]
```

where *indent* is as described in Section 3.13.1.1.

3.13.1.3 The Generalized Itemized List

The generalized itemized list allows the choice of the prefix mark which may be more than a single character. It is introduced by the command

```
.IL   mark[;indent]
```

where *mark* will be used as the prefix string and *indent* is as described in Section 3.13.1.1.

This is the command which is actually used by the “BL” and “DL” commands which generate

```
.IL  \*;indent
```

and

```
.IL  -;indent
```

respectively.

3.13.1.4 List Items in Itemized Lists

The text for each item, other than the first, in an itemized list should be preceded by a *single* blank line.

3.13.1.5 Ending an Itemized List

Following the last item in an itemized list, the

```
.LE
```

command should be used.

3.13.2 Enumerated Lists

The two predefined types of enumerated lists are:

1. *Numbered List*. The items are preceded by sequential numbers followed by periods. This is a numbered list.
2. *Alphabetic List*. The items are preceded by sequential capital letters of the alphabet followed by periods.

A generalized enumerated list mechanism is provided to allow a choice of number format, number prefix, and number suffix.

3.13.2.1 Numbered Lists

A numbered list is introduced by the command

```
.NL   [indent]
```

where *indent* is as described in Section 3.13.1.1.

3.13.2.2 Alphabetic Lists

An alphabetic list is introduced by the command

```
.AL   [indent]
```

where *indent* is as described in Section 3.13.1.1.

3.13.2.3 The Generalized Enumerated List

A generalized enumerated list is introduced by the command

```
.EL   prefix;format;suffix;indent
```

where the arguments are

prefix	the character sequence to precede the number
format	the format code, as described below, indicating how the number will be printed
suffix	the character sequence to follow the number
indent	as described in Section 3.13.1.1

The possible format codes are:

1	printed as 1, 2, 3, 4, ...
a	printed as a, b, c, d, ...
A	printed as A, B, C, D, ...
i	printed as i, ii, iii, iv, ...
I	printed as I, II, III, IV, ...

The “EL” command is the one actually used by the “NL” and “AL” commands which generate

```
.EL   ;1;.;indent
```

and

```
.EL   ;A;.;indent
```

respectively.

3.13.2.4 List Items in Enumerated Lists

The text for each item, other than the first, in an enumerated list should be preceded by a *single* blank line.

3.13.2.5 Ending an Enumerated List

Following the last item in an enumerated list, the

```
.LE
```

command should be used.

3.13.3 Verb Lists

A “verb” list is introduced by the command

```
.VL   [indent]
```

where *indent* specifies the number of characters to indent the text of the list items. In the absence of *indent*, the default value of 16 will be used. This default value is kept in the “Vi_Indent” number register and may be changed with the “Set” command as described in Section 2.6.

3.13.3.1 List Items in Verb Lists

Each item in a verb list is introduced by the command

```
.Verb   phrase
```

where *phrase* is the word or phrase to be displayed at the left margin. The text of the item follows the “Verb” command.

3.13.3.2 Ending a Verb List

The last item of a verb list should be followed by the command

```
.LE
```

3.14 Displays

A *display* is a section of text which is not automatically formatted by PDL/81. Vertical white space will be inserted before and after the display and, if the display cannot fit entirely in the space remaining on the current page, a new page will be started before printing the display.

The List commands (Section 3.13) may not be used within a display. Displays may not be nested.

3.14.1 Generalized Displays

A generalized display is introduced by the command

```
.Ds   [indent]
```

where *indent* is the number of characters to indent each line of the display. In the absence of *indent*, a default value of 8 will be used. This default indent value is kept in the "Ds_Indent" number register and may be changed with the "Set" command as described in Section 2.6.

The display is ended by the command

```
.De
```

The text of a display will be set in *unfilled* mode (see Section 3.6).

3.14.2 Boxed Displays

A *boxed display* is started by the command

```
.Bs   [indent]
```

and ended by the command

```
.Be
```

If specified, *indent* gives the number of characters to indent the display; otherwise, the indentation will be controlled by the current value of the "Ds_Indent" number register.

Except for the box, a boxed display behaves just like a normal (i.e., ds/de) display. Namely, the body is set in *nofilled* mode and a new page will be started if the display cannot fit on the current page.

As an example, the following should be started by ".Bs" and ended by ".Be":

```
Here are some lines which should be in a boxed dis-
play:

    Note that the body is set in nofilled mode.
    Thus you can format it with a screen editor
    and what you see is what will be printed,
    unless the lines don't fit within the
    box.
```

3.14.3 Specialized Displays

A one-line "example" display may be printed by the command

```
.Ex   text
```

which is equivalent to

```
.ds
text
.de
```

Thus, the line

```
.ex this is an example
```

will produce the output

```
this is an example
```

A one-line boxed display, of the style frequently used for "command boxes" in this manual, is produced by the command

```
.Cx   text
```

For example, the line

```
.Cx This is an example
```

will produce

```
This is an example
```

Indentation for the output of the "Ex" and "Cx" commands is controlled by the

"Ds_Indent" number register as described in Section 3.14.1.

3.15 Tags and References

Any point in a document can be assigned a *tag* which may be used to refer to that point from other places. A tag is assigned by the command

```
.Tag   name
```

where *name* is the name to be used to refer to this point in the document. Three pieces of information will be automatically associated with each tag:

- The type of the current document section (e.g., Chapter, Section, Appendix);
- The current section number (e.g., 3, 4.6, 5.9.12); and
- The page number of the current output page.

From elsewhere in the document, the text function

```
#{secref;tag}
```

will return the section type and number of the tag as in

```
#{secref;TAGS}
```

The text function

```
#{pageref;tag}
```

will return the page number of the tag prefixed with the word "page" as in

```
page 30
```

Between runs of PDL/81, the current definitions of any tags for the document are kept in an auxiliary file with a file name as described in the *PDL/81 Introduction and Invocation* manual. If the location of a tag differs from one run to the next in such a way that references to the tag might be incorrect, an error message is issued. The incorrect references can then be corrected simply by running the document through PDL/81 again.

If a source file is moved or renamed and if it has an auxiliary file, the auxiliary file should also be moved or renamed.

3.16 Marginal Characters

The command

```
.Mc char
```

will cause the single character *char* to be printed in the right margin of all non-empty output lines until the command

```
.Mc
```

is encountered. This mechanism, when coupled with the use of a preprocessor, allows automatic generation of “change bars” in new versions of a document.

3.17 Inserting Extra Line Spacing

Blank lines can be inserted after each line of filled text by the command

```
.Els   [n]
```

where *n* is the number of blank lines to insert. If *n* is absent, it is taken as zero. This extra line spacing is inserted only following lines printed in filled mode and will not be added to blank lines or to lines resulting from the “Sp” command. In particular, lines within displays, figures, and tables will not be effected by the “Els” command.

As an example,

```
.Els 1
```

will cause the filled text in a document to be printed double-spaced.

4. The ‘‘Manual’’ Document Style

The *manual* document style is used to format manuals and reports. A document produced in this style consists of:

1. A *title page* containing the name of the manual and its form number;
2. A *title page reverse* containing such items as a copyright notice, a restricted rights legend, and the revision history of the manual;
3. A table of contents showing the various chapters and sections of the manual and lists of any figures or tables contained in the manual (these will normally be printed at the end of the document and must be manually moved to the proper position; however, at the cost of additional processing time, they may optionally be printed in the proper position);
4. The body of the manual;
5. Optionally, one or more appendices; and
6. Optionally, an index.

Note that the format of the title page and the contents of the fixed portion of the title page reverse are defined in the distributed PDL/81 data base to illustrate one of many possibilities. The data base definition files may be changed to meet local standards.

All of the general formatting operations described in Chapter 3 may be used in a manual. The remainder of this chapter describes the operations which are specific to the *manual* document style.

4.1 Title Page and Title Page Reverse

The title page is introduced by the command

```
.TitlePage   [form]
```

where *form* is the document form number to be printed in the upper right corner of the title page.

Each line of the document title is entered by a command of the form

```
.Title   text
```

where *text* is the text of the line. Each of the title lines will be printed double-spaced and *flush right* within the cut-out area of the title page.

The title page reverse, if desired, is introduced by the command

```
.Reverse   dates;permission;restriction;colophon
```

where the arguments are:

dates — text to be printed following the word "Copyright" in the copyright notice. It will normally consist of one or more years as in "1981" or "1980 and 1981". If this argument is absent, a copyright notice will *not* be printed.

permission — if this argument is not null, a "permission to copy" notice will be printed, but only if a copyright notice is also printed.

restriction — if this argument is not null, a "Restricted Rights Legend" will be printed.

colophon — if this argument is not null, a "colophon" will be printed.

Anything which follows the "Reverse" command and precedes the first level 1 heading will be formatted in *filled* and *justified* modes (see Section 3.6) and will be placed following the "canned" portions of the page. The entire contents of the title page reverse will be justified to the *bottom* of the page.

4.1.1 Title Page Format

The default format for the title page of a manual uses right-justified title lines which are positioned vertically to fit in a title-box cutout. This is known as title page style 0.

Another title page style, 1, may be established by the command

```
.Set  TtpStyle;1
```

which will produce a title page with the company name and address centered at the top and the document title boxed and centered on the page. The company name and address is the same as that used on design cover pages and comes from the file "uttl.lib" in the library directory.

The particular style to use by default may be established during installation of the processor.

4.2 Running Heads and Feet

The running page heads for the *manual* document style are as shown in this manual. In particular, odd pages display, from left to right:

- the “chapter” type (CHAPTER or APPENDIX)
- the “chapter” number (or letter if an appendix)
- the “chapter” title
- the page number

and even pages display, from left to right:

- the page number
- a brief document title
- the “chapter” type
- the “chapter” number

The brief document title for display on even pages is given by the command

```
.PTitle   title
```

There are no predefined page feet in the *manual* document style. Page feet may be established as desired with the commands described in Section 3.9. Any feet defined prior to the first chapter will be used on the index, table of contents, and lists of figures and tables.

4.3 Numbered Headings

The formatting styles for numbered heading levels (see Section 3.10) are:

1 centered at the top of a new page, capitalized, underscored, with the “OddChap” number register set to cause level 1 headings to begin on an odd-numbered page

2 flush left, capitalized, underscored, and alone on a line

3 flush left, underscored, and alone on a line

4-7 flush left, underscored, followed by a period, and run-in

Level 1 headings are considered to be “chapters” and the others are considered to be “sections” for purposes of the “Tag” command (see Section 3.15).

A collection of appendices may be introduced by the command

```
.Appendices
```

This command will cause a separator page to be produced containing the word "APPENDICES" centered and boxed. The counter for level 1 headings will be reset and from then on, each level 1 heading will introduce a new appendix instead of a new chapter and alphabetic "numbers" will be assigned to the appendices.

4.4 Table of Contents

A table of contents will be automatically generated with its input being derived from the numbered heading commands (see Section 3.10). The number of levels of headings to be placed in the table of contents is contained in the "TocLevel" number register. By default, this number register contains the value "4". The value may be changed with the "Set" command as described in Section 2.6. If it is set to zero, a table of contents will not be produced.

The table of contents is normally printed at the *end* of the document and should be moved to its proper place before the document is published. However, if the "IToc" number register is set non-zero, an extra processing operation will be invoked which will result in the table of contents being printed in the proper place in the document. The number register may be set with the "Set" command (see Section 2.6).

4.4.1 Table of Contents Format

The default table of contents for a manual is arranged so that the chapter titles stand out and the sections are presented in a non-hierarchical fashion. This is known as table of contents style 0.

Another table of contents style, 1, may be established by the command

```
.Set  TocStyle;1
```

which will produce a conventional, hierarchical table of contents.

The particular style to use by default may be established during installation of the processor.

4.5 Double-Sided Printing

The command

```
.DoubleSided
```

will cause the manual to be printed in double-sided mode, even if the "OddChap" number register is set to zero. When the "DoubleSided" command is encountered, the "OddChap" number register is set from the value of the "DefOddChap" number register.

If double-sided mode is established as the default, the command

```
.SingleSided
```

will use single-sided mode for the current document.

4.6 Index Generation

Entries can be placed into the document index by the command

```
.Ix   text
```

where *text* is the word or phrase to be entered in the index. If a word or phrase is to be indexed and appears at several points in the document, an “Ix” command is required at each point. In other words, the *formatting* of the index is automatic – the *collection* of the entries is not.

When the index is printed, the first character of each entry will be promoted to upper case if it is a lower-case letter. The index sort does not consider the case of letters to be significant.

4.6.1 Additional Indexing Commands

By default, the first word of each index entry is automatically capitalized but this may be changed. The capitalization is under control of the *IndexCap* number register. If it is zero, capitalization is not performed and the items are printed as they were entered. If it is non-zero, capitalization is performed. The setting can be changed in the style file to effect a permanent change. To override the setting for a specific document, use the command

```
.Set  IndexCap;0
```

The command

```
.Ixx  word;string
```

will cause an index entry to be made for “word” which will consist of the word immediately followed by “string”. For example

```
.ixx  customizing;, see tailoring
```

will produce an entry that looks like

```
customizing, see tailoring
```

For simplicity, the command

```
.Ixs  word1;word2
```

is equivalent to

```
.ixx  word1;, see word2
```

4.6.2 Control of Index Printing

If the “Indexing” number register is set to a zero value with the “Set” command (see Section 2.6), an index will not be produced even if indexing commands are encountered.

5. The ‘‘Letter’’ Document Style

The *letter* document style is used to format business letters. They may be printed directly on standard letterhead paper or may be printed on plain paper and photocopied onto letterhead. Note that, as distributed, the formatting of a letter is designed for our standard letterhead. The definition file may be changed to reflect local standards.

The components of a letter are

1. an optional return address;
2. the current date;
3. an optional reference code;
4. the address and salutation;
5. the body;
6. the signature; and
7. optional notations.

All of the general formatting operations described in Chapter 3 may be used in letters. The remainder of this chapter describes operations which are specific to the *letter* style.

5.1 Specific Commands

These commands are used to supply the reference, address, body, signature, and notations portions of the letter. The commands must appear in the order presented.

5.1.1 Specifying the Return Address

If a return address is desired, it may be entered by one or more uses of the command

```
.Return   text
```

where “text” is the text of one line of the return address. For example,

```
.Return Babbage Road
.Return Stevenage
.Return Hertfordshire SG1 2EQ
```

5.1.2 Specifying the Letter Reference

The command

```
.Ref   text
```

is used to supply the reference code of the letter. If this command is used, the text will be placed right justified on a line following the current date at the top of the letter.

5.1.3 Specifying the Address and Salutation

The address and salutation of a letter are introduced by the command

```
.Address
```

This places PDL/81 into *unfilled* mode (see Section 3.6) and positions to the proper place on the output page. The lines for the address and salutation should follow the command.

5.1.4 Specifying the Body of a Letter

The command

```
.Body
```

spaces one blank line, places PDL/81 into *filled* mode, turns off justification so that the right margin will be ragged, and sets the default paragraph indent (see Section 3.7) to zero so as to cause block paragraphs.

The body of the letter should follow this command.

5.1.5 Specifying the Signature of a Letter

At the end of the body, the signature is introduced by the command

```
.Signature
```

which places PDL/81 in *unfilled* mode and positions to the proper place for the signature. The lines of the signature should follow this command.

5.1.6 Specifying Notations for a Letter

Optional notations (e.g., “cc”, “enclosures”) may be placed at the end of the letter with the command

```
.Notations
```

which places PDL/81 in *unfilled* mode and positions to the proper place for the notations. The lines containing the notations should follow this command.

5.2 Running Heads and Feet

The date will be printed in the upper right corner of each page except the first. The page number, enclosed in dashes, will be centered at the bottom of each page except the first. Complete control over page heads and feet is available by use of the commands described in Section 3.9.

5.3 Numbered Headings

The formatting styles for the various levels of numbered headings (see Section 3.10) are:

1 flush left, capitalized, underscored, alone on a line

2 flush left, underscored, alone on a line

3 - 7 flush left, underscored, followed by a period, run-in

The “chapter” type as used by the “Secref” text function is set to “Section” for all levels.

6. The ‘‘Memo’’ Document Style

The *memo* document style is used to format memoranda. They may be printed directly on standard memorandum paper or may be printed on plain paper and photocopied onto memorandum paper. Note that the distributed memo style illustrates one of many possible layouts. The definition file may be changed to reflect local standards.

The components of a memorandum are:

1. header block;
2. distribution list; and
3. body.

All of the general formatting operations described in Chapter 3 may be used in memoranda. The remainder of this chapter describes operations which are specific to the *memo* style. Those relating to the header block and distribution list may be entered in any order but will be printed in a standard order.

6.1 Header Block Commands

The header block of a memorandum consists of the current date, a “from” field, a “ref” field, and a “subject” field. All, except for the date, are optional.

The “from” field is specified by the command

```
.From   text
```

as in

```
.From Stephen H. Caine
```

The “ref” field is specified by the command

```
.Ref   text
```

as in

```
.Ref [SHC/91.221-1]
```

The “subject” field is specified by the command

```
.Subject   text
```

as in

```
.Subject A Sample PDL/81 Memorandum
```

6.2 Distribution List Commands

The distribution list consists of a “to” field and a “cc” field. Both are optional.

The “to” field is specified by the command

```
.To   text
```

as in

```
.To E. Kent Gordon
```

The “cc” field is specified by the command

```
.Cc   text
```

as in

```
.Cc David J. Farber, Jonathan B. Ziegler
```

6.3 The Body of a Memorandum

There is no specific command to introduce the body of a memorandum. The body simply starts with the first line which is *not* a command line. Once the body has started, further header block and distribution list commands are not allowed.

The body is formatted in *filled* and *justified* modes (see Section 3.6) with the default paragraph indentation (see Section 3.7) set to zero.

6.4 Running Heads and Feet

The date will be printed in the upper right corner of each page except the first. The page number, enclosed in dashes, will be centered at the bottom of each page except the first. Complete control over page heads and feet is available by use of the commands described in Section 3.9.

6.5 Numbered Headings

The formatting style for the various levels of numbered headings (see Section 3.10) is:

1 flush left, capitalized, underscored, alone on a line

2 flush left, underscored, alone on a line

3 - 7 flush left, underscored, followed by a period, run-in

The “chapter” type as used by the “Secref” text function is set to “Section” for all levels.

7. The ‘‘Text’’ Document Style

The *text* document style is a “catch all” style for processing text of an unspecified nature. All of the general formatting operations described in Chapter 3 may be used. The remainder of this chapter describes the operations which are specific to the *text* style.

7.1 Body of a “Text” Style Document

The body of a *text* style document is formatted in *filled* and *justified* mode (see Section 3.6) with a default paragraph indent of 5 (see Section 3.7).

7.2 Running Heads and Feet

The date will be printed in the upper right corner of each page. The page number, enclosed in dashes, will be centered at the bottom of each page. Complete control over page heads and feet is available by use of the commands described in Section 3.9. In addition, the command

```
.HStart   n
```

specifies that running heads are to appear starting with page *n* (instead of with page 1), and the command

```
.FStart   n
```

specifies that running feet are to appear starting with page *n* (instead of with page 1).

7.3 Numbered Headings

The formatting style for the various levels of numbered headings (see Section 3.10) is:

1 centered, capitalized, underscored, top of new page

2 flush left, capitalized, underscored, alone on a line

3 flush left, underscored, alone on a line

4 - 7 flush left, underscored, followed by a period, run-in

8. Figures and Tables

The commands described in this chapter allow figures and tables to be placed in a document. PDL/81 treats figures and tables in the same manner, except that:

- The caption for a *figure* is placed at the bottom of the figure while the caption for a *table* is placed at the top of the table;
- Figures and tables are numbered independently; and
- In the "manual" style, references to *figures* are collected in a List of Figures while references to *tables* are collected in a List of Tables.

A figure or table may be defined at any convenient point in the document except within a display, a figure, or a table. It will be printed at the point of definition or, if it is too long to fit on the current page, at the top of the next page.

8.1 Commands for Figures and Tables

Figures and tables may be defined with their body text contained in the source document or they may be defined to leave space for later paste up.

8.1.1 Figures and Tables With In-Source Bodies

A figure is started by the command

```
.Fig   caption[;tag]
```

and ended by the command

```
.EFig
```

while a table is started by the command

```
.Table   caption[;tag]
```

and ended by the command

```
.ETable
```

For both figures and tables, *caption* is the text of the caption and *tag* is an optional tag by which the figure or table may be referenced (see Section 8.4).

The caption of a figure or table is generated from the caption text given in the ".Fig" or ".Table" command and will have the form

```
Fig. nnn  caption-text
```

for figures and

```
Table nnn  caption-text
```

for tables. The font in which captions are printed may be specified as described in Appendix E.

The body of a figure or table may contain any desired information. The ".Fig" or ".Table" command places PDL/81 in *nofilled* mode and the ".EFig" or ".ETable" command returns PDL/81 to its previous mode.

As an example, Fig. 8.1 shows the general format of a figure definition.

```
.Fig text for caption; tag for reference
((                                ))
((                                ))
((    lines for figure body       ))
((                                ))
((                                ))
.EFig
```

Fig. 8.1 Format of a Figure Definition

In PDL/81, that figure could be generated with the commands shown in Fig. 8.2.

```
.Fig Format of a Figure Definition;SAMP-FIG1
  .Fig text for caption; tag for reference
  ((                                 ))
  ((                                 ))
  ((    lines for figure body        ))
  ((                                 ))
  ((                                 ))
  .EFig
.EFig
```

Fig. 8.2 Source for Fig. 8.1

8.1.2 Leaving Space for Later Paste Up

The commands

```
.FigSp   caption[;tag[;space[;art]]]
```

and

```
.TableSp   caption[;tag[;space[;art]]]
```

reserve space for figures and tables, respectively. In these commands, *caption* and *tag* are as described in Section 8.1.1, *space* is the denominator of the fraction

```
1/space
```

which gives the fraction of a full page to be reserved for the figure, and *art* is text which will be printed horizontally and vertically centered in the area reserved for the figure. The *art* text is commonly used to provide an "art" number to be used during paste up.

For example,

```
.FigSp Sample Figure for Paste Up;fg-pu;2;[art: 1234]
```

would generate a half-page figure.

8.2 Figure and Table Numbering

Within a document, figures and tables are numbered independently. The style of numbering is controlled by the "FgnStyle" number register. If it has the value "0", figures and tables are numbered sequentially within the entire document; if it has the value "1", they are numbered sequentially within each chapter and the chapter number is prefixed to the figure or table number.

The default setting for "FgnStyle" is "1" for the *manual* style and "0" for the other styles. If the default is incorrect for a particular document, the value of "FgnStyle" may be changed with the "Set" command prior to the definition of the first figure or table.

8.3 Lists of Figures and Tables

A List of Figures and a List of Tables may be automatically generated when using the “manual” style. Generation of the List of Figures is controlled by the “FglLevel” number register and of the List of Tables by the “TblLevel” number register. If the value is “0”, the corresponding list will not be generated; if it is “1”, the corresponding list will be generated.

If generated, these lists will appear following the Table of Contents and their placement, like that of the Table of Contents, will be under control of the “IToc” number register. If the “OddChap” number register has a non-zero value (specifying that chapters are to start on odd numbered pages), the lists will also be forced to start on odd numbered pages if the “OddList” number register has a value of “1”. If “OddList” has a value of “0”, the lists will not be forced to appear on odd numbered pages.

Each of these number registers has a default value of “1”.

8.4 Figure and Table Referencing

The tag supplied on a “Fig”, “Table”, “FigSp”, or “TableSp” command is defined in the same way as a tag set by the “Tag” command except that an additional field is supplied which contains the type (“Fig.” or “Table”) and the number. This field may be accessed by the text function

```
#{FigRef;tag}
```

9. Security Banners and Portion Marking

Documents printed in both the “text” and the “manual” styles may be supplied with security banners. In addition, documents printed in the “manual style” may be processed in a mode which supports portion marking as specified in DOD 5220.22-M, *Industrial Security Manual for Safeguarding Classified Information*.

9.1 Format of a Classified Document

Each sheet of a classified document will have top and bottom banners which contain the classification level, an optional project name, and a sheet number. Sheets are numbered sequentially, starting at one, and the numbers are independent of the page numbers used in the body of the document. The last sheet of the document will be marked as such and its banners will contain a count of the total number of sheets.

If the document is processed with a non-zero “OddChap” number register (chapters to start on odd numbered pages), the value of “OddChap” will be forced to “3” which will cause skipped, even numbered pages to be printed. These skipped pages will have banners and will be identified as being intentionally left blank.

9.2 Security Banner Commands

These commands, if used, must appear before the first command which would result in any output.

Security banners are established by the command

```
.Security   classification
```

where *classification* is a word or phrase specifying the security level of the document. The command

```
.Project   text
```

specifies *text* to be a project identification word or phrase to be included in the security banner.

In the “manual” style, security banners will appear on the title page and title page reverse unless the number register “TtpSec” is set to zero before the title page is printed.

WARNING

The security banner mechanism will not function correctly if any *output* line in the document *begins* with the string “~??%@&” This seems extremely unlikely, but this string may be redefined when the “text” and “manual” styles are installed.

9.3 Security Banners Formats

By default, banners will have the classification centered, the sheet number on the right, and the project identification on the left. This mode is known as *Security Style* 0.

An alternate security style, 1, is the same as zero except that the sheet number will appear on the left and the project identification will appear on the right on even numbered sheets. This is useful when printing duplexed documents.

Finally, security style 2 produces banners in the same form as produced by Version 1.6 of PDL/81. In this form, the classification will be centered at the top and bottom of each page, the project identification will appear on the left in the top banner and the right in the bottom banner, and the sheet number will appear on the right in the top banner and the left in the bottom banner.

The default security style may be changed by editing the style files. On a per-document basis, the command

```
.Set SecStyle;number
```

where the number is one of the security styles (0, 1, or 2) will set the security style.

9.4 Portion Marking

Both the *manual* style and the new *man2167* style can be run in a mode which supports *portion marking* of classified documents as specified in DOD 5220.22-M, {it Industrial Security Manual for Safeguarding Classified Information}. In this mode, the user sets the security level to *automatic* and uses commands to specify the sensitivity level of each document portion. Paragraphs, headings, figures, and captions may be classified. The maximum sensitivity of any item on a page will be

used to stamp that page and the maximum sensitivity of any item in the document will be used to stamp the document.

The sensitivity of an item (e.g., a paragraph) will be automatically carried to subsequent pages if the item crosses page boundaries. Both sides of a sheet in a double-sided document will be marked with the higher of the two sensitivities.

9.4.1 Defining Classification Codes and Levels

The library file “security.def” defines the codes, marks, and levels. As distributed, this is a fictitious set which is suitable for experimentation. These may be easily changed to match any normal hierarchical classification scheme. In this file, each level is defined by

```
#{class;code;level;short-mark;long-mark}
```

where

code — is the symbol that you will use in commands to specify this classification.

level — is a positive integer value specifying where this classification fits into the set. The lower the number, the lower the classification. The classification corresponding to UNCLASSIFIED *must* have the value 1.

short-mark — The string to use to mark a portion. This should be short, such as U for UNCLASSIFIED.

long-mark — A string used to show this classification on a page or on the entire document.

The distributed values are:

Code	Level	Short-Mark	Long-Mark
u	1	U	UNCLASSIFIED
l	2	L	LOW
nl	3	NATO-L	NATO LOW
ml	4	ML	MID-LOW
nml	5	NATO-ML	NATO MID-LOW
mh	6	MH	MID-HIGH
nmh	7	NATO-MH	NATO MID-HIGH
h	8	H	HIGH
nh	9	NATO-H	COSMIC HIGH

9.4.2 Enabling Portion Marking

Portion marking is enabled for a document by giving it the classification level of "automatic" with the "Security" command as in

```
.Security  AUTOMATIC
```

A document with this classification will have its actual classification levels controlled by other commands in the document.

When enabling portion marking, it is possible to assert that the document will have a given maximum classification level. If the processed document includes material with a higher level, a warning will be issued. The assertion is made by using the second argument of the "Security" command to specify the code of the maximum level. For example, using the security codes of Section 9.4.1,

```
.Security  automatic;ml
```

asserts that the document will not contain material with a classification level higher than MID-LOW.

9.4.3 Marking of Portions

The security classifications of text sections (paragraphs), headings, figures, tables, and captions may all be separately marked.

9.4.3.1 The Class Command

The command

```
.Class  code
```

specifies that all text following, until the next "class" command, is to have the given classification. *Code* must be one of the codes defined in the "security.def" file (Section 9.4.1). At the start of a document, the code corresponding to UNCLASSIFIED will be in effect.

The current classification level will apply to the contents of any figures or tables.

As an example, the command

```
.class  mh
```

will specify that the following material is classified MID-HIGH.

9.4.3.2 Classification of Headings

Headings (e.g., chapter, section, paragraph) are assumed to be UNCLASSIFIED unless explicitly stated otherwise. Unclassified headings will not be marked unless there is at least one heading in the document which *is* classified. In that case *all* headings will be marked.

A heading is marked by giving the classification code as the second argument of the heading command as in

```
.Hn  text;code
```

where *n* is the heading level. For example, a HIGH level four paragraph with a MID-HIGH heading might be started by the commands

```
.class  h
.h4 A Sample Heading;mh
```

9.4.3.3 Classification of Document Title

By default, the title of a document is assumed to be UNCLASSIFIED. A title is marked by giving the classification code as the second argument of the title command as in

```
.Title  text;code
```

For example, a MID-LOW title could be indicated by

```
.Title  A Sensitive Title;ml
```

9.4.3.4 Classification of Captions

By default, the caption of a figure or table is assumed to be UNCLASSIFIED. A caption is marked as classified by using a compound caption argument to the Table, Fig, TableSp or FigSp commands. The compound argument has the form

```
{caption-text;code}
```

Thus, a figure might be declared to have a HIGH contents with a MID-LOW caption by the sequence

```
.class  h
.fig {A Sensitive Caption;ml}
```

A. Error Messages

This Appendix lists error messages which may be issued during processing of a document. Error messages are displayed on the standard error file. If applicable, the message will be prefixed with the name of the current input file and the current line number within the file.

A.1 Non-Terminal Error Messages

The error messages described in this section do not cause termination of PDL/81 processing:

- DUPLICATE TAG: <name> – the given name has been previously defined in another “Tag” command.
- INVALID CHARACTER IN LINE – an input line contains an ASCII control character other than “tab” or “newline”.
- NR: NAME IS NOT AN NR – the name used in a “Set” command is the name of something other than a number register.
- ONE OR MORE TAGS WRONG. REPROCESS TO CORRECT THEM. – The current section or page references of one or more tags does not now correspond to the definitions assumed by PDL/81 at the start of the run. If it is desired to have correct references in the text, run the document through PDL/81 again.
- UNBALANCED BRACKETS – the number of unescaped left brackets is not the same as the number of unescaped right brackets within a call on a text function.
- UNDEFINED TAG: <name> – the given name was referenced in a “Secref” or “PageRef” text function but did not occur also in a “Tag” command.
- UNKNOWN COMMAND – a command name on a command line is not one of those recognized by PDL/81.

A.2 Terminal Error Messages

The error messages described in this section cause immediate termination of PDL/81 processing:

- CAN'T OPEN TEMP FILE <file name> – the named temporary file cannot be opened. This usually means that disk space is not available for the file or that write access privileges are not available in the directory on which the file is to be written.
- CANNOT ALLOCATE DYNAMIC MEMORY FOR A BUFFER – Memory was needed for an input/output buffer, but insufficient memory was available.
- DYNAMIC MEMORY OVERFLOW (n) – all available dynamic memory is allocated and more is needed. The character "n" indicates the particular point in the processor where overflow was detected and is of interest only to PDL/81 processor maintenance personnel.
- MKTEMP: CANNOT GENERATE UNIQUE FILE NAME: <file name> – Names of PDL/81 temporary files are generated by the internal PDL/81 "mktemp" function. This function can generate up to 26 unique names for each invocation of PDL/81. Since names will be reused when possible, and since PDL/81 deletes temporary files after they are closed, this message usually means that a large number of temporaries were left around following a system crash. Examine the directory given in the message and delete the abandoned temporaries.
- UNABLE TO OPEN FILE <file name> – the named file cannot be opened for input. Possibly, it doesn't exist.
- UNKNOWN DEVICE TYPE: <name> – the named device type was specified by an invocation option but no such device is supported.
- UNKNOWN INVOCATION OPTION: <option> – the given option was not recognized by PDL/81.

A.3 Other Error Messages

The error messages described above are those which relate to processing using the document language of PDL/81. Other messages may be issued but they relate to internal processing errors or system problems and should not appear when processing documents. A more complete list of such messages may be found in the *PDL/81 Format Designers Guide.*

B. List of Commands

Address	specify address of a letter
AL	start an alphabetic list
Appendices	start the “Appendices” part of a manual
Appendixes	same as “Appendices”
BL	start a bullet list
Body	start body of a letter
Br	force a line break
Caution	start a “Caution” note
Cc	specify “cc” field of a memo
Class	specify security level of a portion of a manual
Cx	format a command box
Date	set date for document in place of current date
De	end a display
DL	start a dashed list
DoubleSided	print a manual in double-sided form
Ds	start a display
Ef	specify even page feet
EFig	end a figure
Eh	specify even page heads
Eject	start a new page
EL	start a generalized enumerated list
Els	specify extra line spacing
ENote	end a note

ETable	end a table
Ex	format an “example” line
Fig	start a figure
FigSp	define space for later figure paste-up
Fill	enter filled mode
From	specify “from” field of a memo
FStart	specify starting page for feet in “text” style
H1	level 1 heading
H2	level 2 heading
H3	level 3 heading
H4	level 4 heading
H5	level 5 heading
H6	level 6 heading
H7	level 7 heading
Heading	unnumbered heading
HStart	specify starting page for feet in “text” style
IL	start a generalized itemized list
Include	include from an alternate source file
Ix	make an index entry in a manual
Ixx	make a two-part index entry in a manual
Ixs	make a simplified two-part entry in a manual
Justify	justify margins
LE	end a list
Mc	define marginal character
MajorHeading	unnumbered major heading
Need	require a certain number of lines
NL	start a numbered list
NoFill	enter unfilled mode
NoJustify	do not justify margins
Notations	start notations in a letter
Note	start a note
Of	specify odd page feet
Oh	specify odd page heads
Pf	specify page feet
Ph	specify page heads

Project	specify project portion of security banner
Ptitle	specify brief title for a manual
Ref	specify reference code for a letter or memo
Return	specify return address for a letter
Reverse	start title page reverse of a manual
Security	specify security banners
Set	set a number register
Signature	start signature block of a letter
SingleSided	print a manual in single-sided form
Sp	insert vertical white space
SubHeading	unnumbered minor heading
Subject	specify “subject” field of a memo
Table	start a table
TableSp	define table space for later paste-up
Tag	define a text tag
Title	format a manual title line
TitlePage	start title page of a manual
To	specify “to” field of a memo
Verb	specify a “verb” in a verb list
VL	start a verb list
Warning	start a “Warning” note

C. List of Text Functions

Bf	set some text in bold face
Bfu	set some text in bold face with non-blank characters underscored
Bfuc	set some text in bold face with all characters underscored
Cap	capitalize some text
Date	current date as 20 September 1986
Date3	current date as 20 Sep 86
FigRef	figure or table number of a tag
Page	current page number
PageRef	page number of a tag
SecRef	section number of a tag
Time	time of day in form 10:15:20
Time4	time of day in form 10:15
Uc	underscore all characters of some text
Us	underscore non-blank characters of some text

D. List of Number Registers

.NoFF	set to “1” to prevent PDL/81 from generating formfeed control characters in the output; set to “0” to allow PDL/81 to generate formfeeds
.NoTab	set to “1” to prevent PDL/81 from generating horizontal tab control characters in the output; set to “0” to allow PDL/81 to generate tabs
.NoBs	set to “1” to prevent PDL/81 from generating backspace characters in the output; set to “0” to allow PDL/81 to generate backspace characters
.Po	page offset
CapLevel	highest numbered heading level to capitalize
Ds_Indent	display indentation
EjLevel	highest numbered heading level to start a new page
FglLevel	whether or not to generate a list of figures
FgnStyle	style for figure and table numbering
IndexCap	set to “0” to suppress capitalization of first letter in index entry; set to “1” to capitalize
Indexing	switch to specify indexing or not
IToc	switch to specify that table of contents is to be printed in its proper place
Li_Indent	indentation for itemized and enumerated lists
Nt_Indent	indentation for notes
OddChap	switch to put chapters on odd numbered pages and specify style for skipped pages
OddList	whether or not lists of figures and tables should start on odd pages
PDepth	page depth, in lines, to override standard value for specified output device
Pp_indent	paragraph indentation

RiLevel	lowest numbered heading level to be run-in
SecStyle	set to “0” for standard security banner style; set to “1” for duplexed security banner style; set to “2” for old security banner style
Show	set to “0” to suppress progress reports on the standard error file; leave non-zero to provide progress reports
TblLevel	whether or not to generate a list of tables
TocLevel	highest numbered heading level to put in table of contents
TocStyle	set to “0” for non-hierarchical table of contents; set to “1” for hierarchical table of contents
TtpStyle	set to “0” for title page cut-out format; set to “1” for block format
Vi_Indent	indentation for verb lists

E. Changing Various Font Defaults

All of the document language styles define the special fonts:

1 underscored

2 bold face

3 bold face, underscored

The manner in which each of these is accomplished depends on the particular output device being used. Not all devices support all fonts.

Number registers are provided to control the font in which various items in a document are to be printed. Each of these represents a "base" font for the corresponding item and that font may be overridden by using a font change function when the content of an item is defined. The number registers are:

PH_Font	base font for page headers
PF_Font	base font for page footers
FgN_Font	base font for the number portion of a figure or table caption
FgT_Font	base font for the text portion of a figure or table caption
Hn_Font	base font for numbered heading level "n" (n = 1, 2, ..., 7). For the unnumbered headings, a "MajorHeading" will use the level 1 font, a "Heading" will use the level 2 font, and a "SubHeading" will use the level 3 font.

The default settings of these number registers are shown below. Values are interpretted as:

n < 0	use font *abs(n)*, applying it to all characters
n = 0	use the base font
n > 0	use font *n*, applying it to all non-blank characters

They may be overridden for a single document by use of the "Set" command. They may be changed permanently when the various styles are installed.

Number Register	Value	Meaning
PH_Font	0	no special font for page headers
PF_Font	0	no special font for page footers
FgN_Font	0	no special font for caption numbers
FgT_Font	0	no special font for caption text
H1_Font	-1	continuous underscore for level 1
H2_Font	-1	continuous underscore for level 2
H3_Font	-1	continuous underscore for level 3
H4_Font	-1	continuous underscore for level 4
H5_Font	-1	continuous underscore for level 5
H6_Font	-1	continuous underscore for level 6
H7_Font	-1	continuous underscore for level 7

F. Sample Document Source

This appendix presents two sample document language source files. The first shows a typical root file for a multi-part document. The second shows one chapter of that document.

F.1 Sample Root File

```
.*- -smanual
.*
.********** PDL/81 DOCUMENT LANGUAGE REFERENCE GUIDE **********
.*
.*
.include boiler.plt
.PTitle PDL/81 DOCUMENT LANGUAGE REFERENCE GUIDE
.TitlePage 9101-3(3)
.Title PDL/81
.Title DOCUMENT LANGUAGE
.Title REFERENCE GUIDE
.Title #{date}
.Reverse 1981;1;1;1
.include intro.m
.include geninfo.m
.include genform.m
.include manstyl.m
.include ltrstyl.m
.include memstyl.m
.include txtstyl.m
.include figs.m
.include security.m
.appendices
.include errors.m
.include comlist.m
.include funclist.m
.include nrlist.m
.include fonts.m
```

F.2 Sample Chapter Source

```
.*
.h1 General Information
.tag GENINFO
.ix general information
.ix information, general
This chapter discusses aspects of the various PDL/81 document
language styles
```

```
which are of general interest.
It includes information on overall operation of the processor,
general input considerations, and the syntax of document language
commands and functions.
.h2 Overall Operation
.ix overall operation
.ix operation overall
PDL/81 processes a document in one pass through the source.
If a table of contents is requested, it will normally be printed
at the #{us;end} of the document and can be moved to its proper
place for publication.
At the expense of additional processing time, however, the table of
contents may be printed in-line under control of the "IToc"
number register
as described in #{secref;toc}.

During operation, PDL/81 maintains a dictionary of #{us;tags}
which are
used in referring to some point in a document from some other point
(see #{secref;tags}).
At the end of a run, the dictionary is written to an auxiliary file
which will be read back in the next time the document is processed.
.h2 Input Format
.ix input format
.ix format of input
Input to PDL/81 consists of a sequence of source lines.
Each line is terminated by a newline character.
The only ASCII control codes
.ix ASCII control codes
.ix control codes, ASCII
allowed are "tab" and "newline".
This section describes the interpretation of various special
characters
and character sequences within source lines.
.h3 Tab Expansion on Input
.ix tab expansion
.ix expansion of tabs
ASCII tab characters are allowed on input lines.
Each tab will be replaced by enough blanks to position the
immediately
following character to the next input tab stop.
Input tab stops are set at columns 1, 9, 17, ....
.h3 Continuation of Input Lines
.ix continuation of input lines
.ix input line continuation
The sequence "\\<newline>" results in deletion of both
characters, thus causing
the following line to be considered part of the current line.
The character "\\" is known as the #{us;escape character} and has
.ix escape character
.ix \\\\ as escape character
additional uses as described throughout this manual.
.h3 Special Characters
.ix special characters
The escape character
.ix escape character
may be used to input two special characters -- the
"bullet" and the
"unpaddable space".
.h4 The Bullet Character
.ix bullet character
.ix \\\\* as bullet character
.ix escape character
The special sequence "\\*" will be replaced by the
so-called #{us;bullet}
character ("\*") in the output.
```

```
This character is normally formed by superimposing
the letter "o" and the
character "+".
.h4 The Unpaddable Space
.ix unpaddable space
.ix space, unpaddable
.ix blank, unpaddable
.ix escape character
The escape character
followed by a space is known as the #{us;unpaddable space}.
It will be replaced by a single space in the output but will not be
subject to size expansion during justification and will not act as a
word break.
.h2 Command Lines
.ix command lines
If the first character of a line is a period (".),
.ix . as command character
the line is known as a #{us;command line}.
Command lines contain #{us;commands} which direct various types of
processing or provide information to PDL/81.

When a command line is encountered, white space
(blanks and tabs) following
the initial period is skipped.
If a newline is encountered, the command line is ignored.
If an asterisk ("*") is encountered, the line is considered to be a
#{us;comment command},
.ix comment command
.ix command, comment
the rest of the line is skipped, and the whole line is ignored.

If anything else is encountered, it is assumed
to start a #{us;command name}
.ix command name
.ix name of a command
which extends to the first blank, tab, or newline.
After skipping any white space, the remainder of
the line, if any, is
taken as one or more #{us;arguments} of the command.
.ix arguments of command
.ix command arguments
Arguments are separated from each other by semicolons (";").
Thus, the general form of a command line is
.cx .name [argument[\;argument]...]
where the brackets indicate optional material.

For example,
.ex .Space   5
is a command line with a command name of "Space"
and an argument of "5".

The case (upper, lower, mixed) of a command name is immaterial.
Thus, "Space", "SPACE", "space", or even "sPAce"
all represent the same
command name.

If a command argument includes a semicolon, the
semicolon must be protected
by an escape character
.ix escape character
so that it will not be taken as an argument separator.
For example, the command
.ex .PTitle arg with \; in it
should be entered as
.ex .PTitle arg with \\\; in it
if the single argument is to be "arg with ; in it".
```

```
If an input line beginning with a period is not
to be taken as a command line,
the period must be protected with an escape character.
.ix escape character
For example, the input line
.ex . line with a period as first character
should be entered as
.ex \\. line with a period as first character
.h2 Text Functions
.ix text functions
.ix functions, text
Text functions are used to insert special
information into a document or to
perform some kind of textual modification.
The general form of a text function invocation is
.cx \#{name[\;argument[\;argument]...]}
where the brackets indicate optional information.
The case (upper, lower, mixed) of a function name is immaterial.
If an argument to a text function contains any of
.ex \#{   \{   \}   \;
each must be preceded by an escape character.
.ix escape character
Thus, to invoke function "func" with one
argument being "\#{" and another
being ";;",
.ds
\#{func;\\\#{;\\;\\;}
.de
may be used.

The entire text of a function invocation
must be contained on a single
(possibly continued) source line.
Within a function call, leading white space
on a continuation line is ignored.
.h3 Nesting of Function Calls
.ix nesting of function calls
.ix function call nesting
Calls on text functions may be nested.
For example, consider the "us" function which
underscores its argument
and the "cap" function which changes each lower-case
letter in its argument
to upper case.
The sequence
.ex \#{us\;\#{cap\;this is a test}}
or
.ex \#{cap\;\#{us\;this is a test}}
would result in
.ex #{us;#{cap;this is a test}}
.h2 Including Alternate Source Files
.ix alternate source files
.ix source files, alternate
.ix including alternate source files
At any point in a source file, input may be switched to
another source file
by the command
.cx .Include   file
.ix .Include command
.ix Include command
where #{us;file} is the name of the file to be included.
Files included with an "Include" command may contain
"Include" commands.
.h2 Number Registers
.tag NRS
```

```
.tag SET
.ix number registers
A #{us;number register} is a numeric-valued variable used to provide
information to PDL/81.
The various number registers used in the document language
are discussed
throughout this manual.

A number register may be assigned or reassigned a value by
the command
.cx .Set   name\;value
.ix Set command
.ix .Set command
where #{us;name} the name of the number register
and #{us;value} is a number giving the value.

For example,
the ".po" number register contains
.ix .po number register
the current page offset.
Within a document, it may be set by the "Set" command as in
.ex .Set   .po\;12
which would establish the current page offset as twelve character
positions.

The command
.cx .Incr   name;value
.ix .Incr command
.ix Incr command
will the increment the named number register by the
given value, and the
command
.cx .Decr   name;value
.ix .Decr command
.ix Decr command
will decrement the named number register by the given value.
.h3 Global Control Number Registers
.ix global control number registers
.ix number registers, global control
Several number registers can be used to control
various global aspects
of a document as described in this section.
.h4 Page Offset
The global page offset is contained in the ".po" number register.
.ix page offset
.ix offset, page
.ix .po number register
The value of this register is the number of character
positions by which each
output line is to be indented, thus shifting the entire
document to the
right on the output page.
The default value of the ".po" number register depends on
the selected device.
A new value may be assigned by
.cx .Set   .po\;value
.h4 Use of Output Formfeed Characters
The ".NoFF" number register
.ix .NoFF number register
is a switch which
specifies whether or not formfeed control characters
may be inserted in the
output by PDL/81.
If the value of the number register is zero, PDL/81 may
insert formfeeds.
.ix formfeed control character
```

```
If the
value of the number register is non-zero, PDL/81 will not
insert formfeeds.
The default value of the number register depends on the
selected device.
It may be changed by the command
.cx .Set   .NoFF\;value
.h4 Use of Output Tab Characters
The ".NoTab" number register
.ix .NoTab number register
is a switch which
specifies whether or not horizontal tab
control characters may be inserted in the
output by PDL/81.
If the value of the number register is zero, PDL/81 may insert tabs.
.ix tab control character
If the
value of the number register is non-zero, PDL/81 will not insert tabs.
The default value of the number register depends on the
selected device.
It may be changed by the command
.cx .Set   .NoTab\;value
.h4 Use of Output Backspace Characters
The ".NoBs" number register
.ix .NoBs number register
is a switch which specifies whether or not backspace characters
.ix backspace characters
may be inserted in the output by PDL/81.
If the value of the number register is zero, PDL/81 may
insert backspace
characters.
If the value of the number register is non-zero, PDL/81
will not insert
backspace characters and operations such as overstriking
and underlining
will be performed by other means.
The default value of the number register depends on the
selected device.
It may be changed by the command
.cx .Set   .NoBs\;value
.h4 Controlling Progress Reports
.ix controlling progress reports
.ix progress reports
The "Show" number register
.ix Show number register
is a switch which controls whether or not a report of
processing progress
is to be displayed on the standard error file.
.ix standard error file
If the value of the number register is zero, a report
is not displayed.
If the value is non-zero, a report is displayed.
The default value is #{show}.
It may be changed by the command
.cx .Set   Show\;value
```

Index