the tool for software designers

PDL/81™

Software User's Manual
2167 Manual Style

(Version 2.0)

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: 9102-51

1 August 1988

Preface

This manual describes the PDL/81 document style known as *man2167*. This style is designed to aid in producing documents that must abide by the guidelines of DOD-STD-2167, *Defense System Software Development*, and MIL-STD-490A, *Specification Practices*. It should be particularly useful in producing the various documents specified in Data Item Descriptions DI-MCCR-8009 through DI-MCCR-8032.

The remainder of this manual is formatted by the PDL/81 processor using the *man-2167* style and, thus, provides a detailed example of the style.

NOTE

This is an entirely UNCLASSIFIED document. It contains NO classified material. Classification markings, declassification instructions, and warning notices, if any, are for purposes of illustration only.

9101-51(0)

SOFTWARE USER'S MANUAL

FOR THE

PDL/81 2167 MANUAL STYLE

CONTRACT NO. ABC123-87-R-A000

CDRL SEQUENCE NO. 00001A

30 May 1988

Prepared for:

PDL/81 USERS

Prepared by:

CAINE, FARBER & GORDON, INC.
1010 EAST UNION STREET
PASADENA, CA 91106

RESTRICTED RIGHTS LEGEND

Comments or questions relating to this manual or to the subject software should be addressed to Caine, Farber & Gordon, Inc.; 1010 East Union Street; Pasadena, California 91106; Telephone (818) 449-3070; Telex 295316 CFG UR.

This manual was formatted by PDL/81.

CONTENTS

FIGURES

TABLES

APPENDICES

APPENDIX I

APPENDIX II

APPENDIX III

APPENDIX IV

APPENDIX V

[THIS PAGE INTENTIONALLY LEFT BLANK]

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 data base which is used to tailor the processor to the particular requirements of the document being produced. As distributed, the data base includes definitions for such document styles as:

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

This manual describes a special document style known as "Man2167" which is intended to aid in formatting documents which must abide by the guidelines of DOD-STD-2167, Defense System Software Development, and MIL-STD-490A, Specification Practices. It should be particularly useful in producing the various documents specified in Data Item Descriptions DI-MCCR-8009 - DI-MCCR-8032.

This manual is formatted in the general style of a 2167 document. This has been done solely for purposes of illustration. Classification notices, contract numbers, CRDRL sequence numbers, etc. are completely fictitious.

Other manuals (see Paragraph 1.3) describe the other data base components and the methods for modifying the data base.

1.1 FEATURES OF THE MAN2167 STYLE. The document language described in this manual provides a number of formatting features including:

- automatic generation of required title page
- automatic line filling and justification
- twelve levels of section and paragraph 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 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 formated displays

- collection and formatting of a document index from user-provided index terms

- optional security banners with sheet count control

1.2 DOCUMENT STYLES AND THE PDL/81 DATA BASE. The document styles which are available at an installation reside in the PDL/81 "data base". The form of the data base 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*, Caine, Farber & Gordon, Inc. order number 9101-17

- *PDL/81 Format Designers' Guide*, Caine, Farber & Gordon, Inc. order number 9101-1

- *PDL/81 Design Language Reference Guide*, Caine, Farber & Gordon, Inc. order number 9101-2

- *PDL/81 Document Language Reference Guide*, Caine, Farber & Gordon, Inc. order number 9101-3

- *PDL/81 Invocation and Tailoring Guide*, Caine, Farber & Gordon, Inc. order number 9101-18

NOTE

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 as described in the "PDL/81 Installation Guide".

[THIS PAGE INTENTIONALLY LEFT BLANK]

2. GENERAL INFORMATION

This chapter discusses general aspects of the Man2167 document style. 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 Paragraph 4.3.

2.1.1 Tag Dictionary. 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 Paragraph 3.16). 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 ("*") in the output. This character is normally formed by superimposing the letter "o" and the character "+".

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.

2.3.1 Comment Commands. 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.

2.3.2 Command Name. 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".

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

2.3.3 Use of Semicolons in Command Arguments. 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".

2.3.4 Beginning a Line With a Period. 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.

2.4.1 Continuation Within Text Functions. 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.

2.4.2 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 lower-case 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.

2.6.1 The SET Command. A number register may be assigned or reassigned a value by the command

```
.Set   name;value
```

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

2.6.1.1 Example. For example, the ".po" number register contains the current page offset. Within a document, it may be set by the "Set" command as in

```
.Set   .po;12
```

which would establish the current page offset as twelve character positions.

2.6.2 The INCR and DECR Commands. The command

```
.Incr   name
```

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

```
.Decr   name
```

will decrement the named number register by the given value.

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

2.6.3.1 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
```

2.6.3.2 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
```

2.6.3.3 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
```

2.6.3.4 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
```

2.6.3.5 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. If the value is non-zero, a report is displayed. The default value is 1. It may be changed by the command

```
.Set    Show;value
```

[THIS PAGE INTENTIONALLY LEFT BLANK]

3. GENERAL FORMATTING OPERATIONS

This chapter describes a number of general-purpose formatting operations. 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 30 May 1988 and the text function

```
#{date3}
```

returns the date as 30 May 88. Thus, the line which produces

```
        Today is 30 May 1988 (30 May 88)
```

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 17:05:28 and the text function

```
#(time4)
```

returns the time as 17:05. Thus, the line which produces

```
The time is 17:05:28 (17:05)
```

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 Paragraph 3.10). This can be done by the text function

```
#{page}
```

which returns the current page number as its value. For example, the line

```
        This is on page #{page}.
```

will produce

```
        This is on page 15.
```

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

3.4.1.1 Underscored Blanks. 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.

3.4.2.1 Functions. The function are

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

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.

3.6.1 Sentences. 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.

3.6.2 Unfilled Mode. 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.

3.6.2.1 NOFILL and FILL Commands. 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 (Paragraph 3.11).

3.6.3 Turning Justification On and Off. Justification is turned off by the command

```
.NoJustify
```

and is turned on by the command

```
.Justify
```

3.6.4 Forcing a Line Break. 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 0 spaces. The indentation amount is kept in the "Pp_Indent" number register and can be modified with the "Set" command as described in Paragraph 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.

3.8.1 Contiguous SP Commands. 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.

3.9 FORCING NEW PAGES.

3.9.1 The EJECT Command. 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.

3.9.2 Producing Floating Blank Pages. 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 cumulative.

```
        .PF ;- #(page) -
```

Heads and feet are normally printed in the base font. This may be changed, however, as described in Appendix 50.

3.11 NUMBERED HEADINGS. The PDL/81 Man2167 document style supports twelve levels of numbered headings. In keeping with 490A restrictions, however, a warning is issued if more than seven levels are used in a document.

3.11.1 The Hn Command. A numbered heading is set by the command

```
.Hn   text
```

where n is one of the numbers through 12 and text is the text to be used for the heading. Level 1 headings are considered the most important while level 12 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.

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

3.11.1.2 Level Counters. 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.

3.11.1.3 Formatting of Headings. The way in which a given heading level is formatted depends upon installation parameters. By default, the formatting is:

1 Capitalized and centered, at the top of a new page

2 Capitalized, underscored, flush left, followed by a period, run in

3 - 12 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 was set with

.H4 Formatting of Headings

3.11.2 Other Numbered Heading Commands. Managing a deeply nested set of headings can become very complicated. To ease this task, both relative headings and bracketed headings are introduced.

3.11.2.1 Relative Headings. The relative heading command allow setting of headings at a level relative to the current heading level.

3.11.2.1.1 The H Command. The command

```
.H   text
```

establishes a heading at the current level. For example, it is equivalent to an "H3" command if the last heading command was an "H3" command.

3.11.2.1.2 The H+ Command. The command

```
.H+   text
```

establishes a heading at the level next deeper than the current level. For example, it is equivalent to an "H4" command if the last heading command was an "H3" command.

3.11.2.1.3 The H- Command. The command

```
.H-   text
```

establishes a heading at the next outer level. For example, it is equivalent to an "H2" command if the last heading command was an "H3" command.

3.11.2.2 Bracketed Heading Commands. The command

```
.Sec   text
```

establishes a heading at the next deeper heading level. The heading level is restored by the

```
.ESec
```

command.

3.11.3 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 Paragraph 2.6).

3.11.3.1 Changing the Eject Level. 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.

3.11.3.2 Changing Captitalization Level. 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.

3.11.3.3 Changing Run-In Level. 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.

3.11.3.4 Forcing Chapters to Odd Pages. 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.

3.11.3.5 Selecting Heading Fonts. The font in which a given level of heading will be printed may be changed as described in Appendix 50.

3.12 UNNUMBERED HEADINGS. The Man2167 document style supports three styles of unnumbered headings which are completely independent of the numbered heading mechanism described in Paragraph 3.11. Appropriate vertical spacing is performed before and after each of the unnumbered headings.

3.12.1 The MAJORHEADING Command. The command

```
.MajorHeading   text
```

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

3.12.2 The HEADING Command. The command

```
.Heading   text
```

will set text flush left, capitalized, and underscored.

3.12.3 The SUBHEADING Command. The command

```
.SubHeading   text
```

will set text flush left and underscored.

3.13 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.

This note was started by

```
        .Note CAUTION
```

and ended by

```
        .ENote
```

3.13.1 Simplified Commands. For convenience, the command

```
.Caution
```

is equivalent to

```
.Note   CAUTION
```

and the command

```
.Warning
```

is equivalent to

```
.Note   WARNING
```

3.14 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.

3.14.1 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.

3.14.1.1 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 Paragraph 2.6).

3.14.1.2 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 Paragraph 3.14.1.1.

3.14.1.3 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 Paragraph 3.14.1.1.

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

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

and

```
.IL  -;indent
```

respectively.

3.14.1.4 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.

3.14.1.5 Ending an Itemized List. Following the last item in an itemized list, the

```
.LE
```

command should be used.

3.14.2 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.

3.14.2.1 Numbered Lists. A numbered list is introduced by the command

```
.NL    [indent]
```

where indent is as described in Paragraph 3.14.1.1.

3.14.2.2 Alphabetic Lists. An alphabetic list is introduced by the command

```
.AL    [indent]
```

where indent is as described in Paragraph 3.14.1.1.

3.14.2.3 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 Paragraph 3.14.1.1

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.

3.14.2.4 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.

3.14.2.5 Ending an Enumerated List. Following the last item in an enumerated list, the

```
.LE
```

command should be used.

3.14.3 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 Paragraph 2.6.

3.14.3.1 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.

3.14.3.2 Ending a Verb List. The last item of a verb list should be followed by the command

```
.LE
```

3.15 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 (Paragraph 3.14) may not be used within a display. Displays may not be nested.

3.15.1 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 Paragraph 2.6.

3.15.1.1 Ending a Display. The display is ended by the command

```
.De
```

The text of a display will be set in unfilled mode (see Paragraph 3.6.2).

3.15.2 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.

3.15.2.1 Behavior of Boxed Displays. 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.

3.15.2.2 Example. As an example, the following was started by ".Bs" and ended by ".Be":

```
Here are some lines which should appear within a boxed
    display:

    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.
```

3.15.3 *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 Paragraph 3.15.1.

3.16 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

    Paragraph 3.16

The text function

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

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

    page 35

For example, the beginning of the current section contains the command

```
.Tag tags
```

The line

```
See #{secref;tags} on #{pageref;tags}.
```

will produce

    See Paragraph 3.16 on page 35.

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.

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

3.17 MARGINAL CHARACTERS. The command

```
.Mc char
```

will cause the single character char to be pinted 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.

3.18 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.

3.18.1 Example. As an example,

```
.Els 1
```

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

[THIS PAGE INTENTIONALLY LEFT BLANK]

4. DOCUMENT STRUCTURE

A document produced in the Man2167 style consists of:

1. A title page automatically formatted to the required standard;

2. An optional 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 sections and numbered paragraphs 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.

4.1 TITLE PAGE AND TITLE PAGE REVERSE. The title page may be either automatically or manually formatted.

4.1.1 Automatic Title Page Formatting. When automatic formatting is desired, the information to compose the page is taken from a series of commands and/or from information in the PDL/81 data base. The generated format is:

[Document Name]

FOR THE

[Project Name]

CONTRACT NO. [Contract Number]

CDRL SEQUENCE NO. [CDRL Number]

[Date of document (day month year)]

Prepared For:

[Contracting Agency Name, department code]

Prepared by:

centered on the page with the Document Control Number in the upper right corner.

4.1.1.1 <u>Document</u> <u>Control</u> <u>Number</u>. The document control number is established by the command

```
.Control   text
```

4.1.1.2 <u>Document</u> <u>Name</u>. The document name is established by the command

```
.DocName   text
```

If the name should be displayed on several output lines, separate the text for those lines by semicolons.

4.1.1.3 Project Name. The project name is established by the command

```
.Project   text
```

If the name should be displayed on several output lines, separate the text for those lines by semicolons.

4.1.1.4 Contract Number. The contract number is established by the command

```
.Contract   text
```

4.1.1.5 CDRL Sequence Number. The CDRL sequence number is established by the command

```
.CDRL   text
```

4.1.1.6 Agency Name. The contracting agency name is established by the command

```
.Agency   text
```

If the name should be displayed on several output lines, separate the text for those lines by semicolons.

4.1.1.7 Contractor Name and Address. The contractor name and address information is established by the command

```
.Contractor   text
```

If this information should be displayed on several output lines, separate the text for those lines by semicolons.

4.1.1.8 Establishing Default Information. Just before processing a document, a check is made to see of the file "m2167.def" exists in the data-base library. If it does exist, it is input and processed. The file should contain one or more of the commands Control, DocName, Project, Agency, CDRL, Contract, or Contractor. Any of these commands also present in the document source will override those in the man2167.def file.

4.1.2 Manual Title Page Formatting. The title page may be formatted manually by introducing it with the command

```
.TitlePage   [control]
```

where control is the document control number.

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 centered.

4.1.3 Title Page Reverse. 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 "1986" or "1986 and 1987". If this argument is absent, a copyright notice will <u>not</u> 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 <u>filled</u> and <u>justified</u> modes (see Paragraph 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 <u>bottom</u> of the page.

4.2 <u>APPENDICES</u>. If the document contains appendices, the command

```
.Appendices
```

should be placed before the first appendix. Each appendix is then begun with a level-one (H1 command) heading. As required by the standards, the appendices are numbered I, II, III, ... and the paragraphs are numbered 10, 20, 30, ..., respectively. An error message is issued if more than nine level-one sections precede the first appendix.

4.3 TABLE OF CONTENTS. A table of contents will be automatically generated with its input being derived from the numbered heading commands (see Paragraph 3.11). 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 "12". The value may be changed with the "Set" command as described in Paragraph 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 Paragraph 2.6).

4.4 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.

4.4.1 Special Index Entries. The command

```
.ixx  word
```

will cause an index entry to be made for "word" which will consists of the word immediately followed by the string. For example,

```
.ixx  customizing
```

will cause an entry that looks like

```
customizing, see tailoring
```

For simplicity, the command

```
.ixs  words1
```

is equivalent to

```
.ixx  word1
```

4.4.2 Format of Index Entry. When the index is printed, the first character of each entry will normally be promoted to upper case if it is a lower-case letter. This can be prevented by the command

```
Set IndexCap
```

The index sort does not consider the case of letters to be significant.

4.4.3 Suppressing Indexing. If the "Indexing" number register is set to a zero value with the "Set" command (see Paragraph 2.6), an index will not be produced even if "Ix" commands are encountered.

[THIS PAGE INTENTIONALLY LEFT BLANK]

5. 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

* References to *figures* are collected in a List of Figures while references to *tables* are collected in a List of Tables.

5.1 *LOCATION OF FIGURES AND 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 normally 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.

5.1.1 *Collecting Figures and Tables Before Appendices*. The command

```
.Set FigLoc;1
```

will cause all figures to be collected together and printed just before the first Appendix. The command

```
.Set TblLoc;1
```

will cause all tables to be collected together and printed just before the first Appendix.

This placement includes tables or figures which may occur in appendices.

5.1.2 Collecting Figures and Tables at the End of the Document. The command

```
.Set FigLoc;2
```

will cause all figures to be collected together and printed at the end of the document. The command

```
.Set TblLoc;2
```

will cause all tables to be collected together and printed at the end of the document.

5.2 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.

5.2.1 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 Paragraph 5.5).

5.2.1.1 Figure and Table Captions. 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 50.

5.2.1.2 Figure and Table Bodies. 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.

5.2.1.3 Example. As an example, Figure 1. shows the general format of a figure definition.

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

Figure 1. Format of a Figure Definition

It was generated with the commands shown in Figure 2..

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

Figure 2. Source for Figure 1.

5.2.2 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 Paragraph 5.2.1, *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.

5.2.2.1 *Example*. For example,

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

will generate the half-page Figure 3. on page 51 and

```
.TableSp Sample Table for Paste Up;tb-pu;2;[art: 5678]
```

[art: 1234]

Figure 3. Sample Figure for Paste Up

Table I. Sample Table for Paste Up

[art: 5678]

will generate the half-page Table I. on page 52.

5.3 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 settings for "FgnStyle" is .

5.4 LISTS OF FIGURES AND TABLES. A List of Figures and a List of Tables will be automatically generated when using the "man2167" style. These lists will appear within the Table of Contents.

5.5 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}
```

For example, the first figure defined in this chapter has the tag "SAMP-FIG1". Thus, the line

```
See #{FigRef;SAMP-FIG1} on #{PageRef;SAMP-FIG1}
```

will produce

```
See Figure 1. on page 50
```

[THIS PAGE INTENTIONALLY LEFT BLANK]

6. SECURITY BANNERS AND PORTION MARKING

Documents printed in the "man2167" style may be supplied with security banners. In addition, portions of the document (e.g. paragraphs, figures, captions) may be independently marked with the classification level of each page, and of the entire document, being automatically determined.

6.1 FORMAT OF A CLASSIFIED DOCUMENT. Each sheet of a classified document will have top and bottom banners which contain the classification level, sheet number, and total sheet count for the document. 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.

6.2 SECURITY BANNER COMMAND. This command, 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.

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.

6.3 PORTION MARKING SUPPORT. The "man2167" style can be run in a mode which supports portion marking of classified documents as specified in DOD 5220.22-M, 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.

6.3.1 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

6.3.2 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 maximum level. For example, using the security codes of Paragraph 6.3.1,

```
.Security  automatic;1
```

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

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

6.3.3.1 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 . 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.

6.3.3.2 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
```

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

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

6.3.3.3 <u>Classification of Document Title</u>. 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
```

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

```
.Title  A Sensitive Title
```

6.3.3.4 <u>Classification of Captions</u>. 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)
```

[THIS PAGE INTENTIONALLY LEFT BLANK]

APPENDIX I
10. 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.

10.1 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.

10.2 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.

10.3 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 manual "PDL/81 Format Designers' Guide" (see Paragraph 1.3).

APPENDIX II
20. LIST OF COMMANDS

Agency	specify contracting agency
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
CDRL	specify CDRL sequence number
Contract	specify contract number
Contractor	specify contractor name and address
Control	specify document control number
Cx	format a command box
Date	set date for document in place of current date
De	end a display
DL	start a dashed list
DocName	specify document name
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
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
IL	start a generalized itemized list
Include	include from an alternate source file
Ix	make an index entry in a manual

Ixs	make a "see" index entry
Ixx	make a special index entry
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
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 name
Reverse	start title page reverse of a manual
Security	specify security banners
Set	set a number register
Sp	insert vertical white space
SubHeading	unnumbered minor heading
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
Verb	specify a "verb" in a verb list
VL	start a verb list
Warning	start a "Warning" note

APPENDIX III
30. 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 30 May 1988
Date3	current date as 30 May 88
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 17:05:28
Time4	time of day in form 17:05
Uc	underscore all characters of some text
Us	underscore non-blank characters of some text

[THIS PAGE INTENTIONALLY LEFT BLANK]

APPENDIX IV
40. 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
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
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

Show set to "0" to suppress progress reports on the standard error file; leave non-zero to provide progress reports

Vi_Indent indentation for verb lists

APPENDIX V
50. CHANGING VARIOUS FONT DEFAULTS

The "man2167" document language style defines 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.

HnN_Font font for the numbers that start numbered heading level "n".

HnS_Font font for the space that separates the number and the text for numbered heading level "n".

The default settings of these number registers are shown in Table II.. 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, aplying 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.

Table II. Default Document Font Settings

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	0	no special font for level 1
H2_Font - H12_Font	1	underscore for levels 2 - 12
H1N_Font - H12N_Font	0	no special font for heading numbers
H1S_Font - H12S_Font	0	no special font for heading spaces

INDEX

LAST SHEET OF DOCUMENT