VAX-11
DIGITAL Standard Runoff (DSR) User's Guide

Order No. AA-J268B-TK

May 1982

This manual documents the DIGITAL Standard Runoff (DSR) text-formatting utility.

REVISION/UPDATE INFORMATION:	This revised document supersedes the DEC Standard Runoff (DSR) User's Guide (Order No. AA-J268A-TK).
OPERATING SYSTEM AND VERSION:	VAX/VMS Version 3.0
SOFTWARE VERSION:	DIGITAL Standard Runoff Version 2.0

digital equipment corporation · maynard, massachusetts

First Printing, November 1979
Revised, May 1982

DEC	DIBOL	RSX
DEC/CMS	EduSystem	UNIBUS
DECnet	IAS	VAX
DECsystem-10	MASSBUS	VMS
DECSYSTEM-20	PDP	VT
DECUS	PDT	digital
DECwriter	RSTS	

ZK2196

CONTENTS

CONTENTS

Page

CONTENTS

EXAMPLES

TABLES

PREFACE

MANUAL OBJECTIVES

This manual describes the text-formatting program, DIGITAL Standard Runoff (DSR), and provides usage and reference information on the program.

INTENDED AUDIENCE

This manual is intended for users of the VAX/VMS operating system who need to format documents. Users are expected to have some familiarity with VAX/VMS system concepts and to know how to use a text editor (such as EDT).

STRUCTURE OF THIS DOCUMENT

This manual contains six chapters and four appendixes.

- Chapter 1, Introduction to DSR, provides an overview of DSR, describes terms and conventions used in DSR, and gives some simple examples to introduce the user to a few DSR commands and flags.

- Chapter 2, The DSR Commands, describes all of the DSR commands.

- Chapter 3, DSR Flags and Flag Control Commands, describes all of the DSR flags and flag control commands.

- Chapter 4, Creating a Table of Contents, describes how to create a table of contents using table-of-contents commands and the Table of Contents (TOC) program.

- Chapter 5, Creating an Index, describes how to create an index using indexing commands and the Two-Column Index (TCX) program.

- Chapter 6, Running DSR, explains how to run DSR and describes all of the command line qualifiers.

- Appendix A, DSR/RUNOFF Compatible Features, describes commands from earlier versions of RUNOFF that are compatible with DSR.

- Appendix B, DSR Messages, contains a list of DSR error messages (including informational messages) and gives an explanation and user action for each one.
- Appendix C, DSR Command Summary, gives a concise list of all of the DSR commands and flags and their legal abbreviations.
- Appendix D, Template Samples, contains two sample templates in which DSR commands and flags are used.

ASSOCIATED DOCUMENTS

- The VAX-11 DIGITAL Standard Runoff Pocket Reference Guide provides the more experienced user with concise explanations of DSR commands, flags, and the TOC and TCX programs.
- The VAX-11 Information Directory and Index contains a complete list of all VAX-11 documents and a master index of all topics referred to in the VAX/VMS documentation set.

CONVENTIONS USED IN THIS DOCUMENT

Convention	Meaning
UPPERCASE WORDS AND LETTERS	Uppercase words and letters used in examples indicate that you should type the word or letter exactly as shown.
lowercase words and letters	Lowercase words and letters used in format examples indicate that you are to substitute a word or value of your choice.
quotation marks apostrophes	The term "quotation marks" refers to double quotation marks ("). The term "apostrophe" refers to a single quotation mark (').
[ ]	Square brackets indicate that the enclosed item is optional.
{ }	Braces are used to enclose lists from which one element is to be chosen.
...	A horizontal ellipsis indicates that the preceding item(s) can be repeated one or more times. For example: `file-spec[,file-spec...]`
. . .	A vertical ellipsis indicates that not all of the statements in an example or figure are shown.
$ RUN $_File:	All user input in examples is printed in red ink. All output that the system prints or displays is printed in black ink.

Convention	Meaning
RET	A symbol with a 1- to 3-character abbreviation indicates that you press a key on the terminal.
CTRL/X	CTRL/x indicates that you must hold down the key labeled CTRL while you press another key, for example, CTRL/C, CTRL/Y, or CTRL/Z. In examples, this control key sequence is shown as ^C, ^Y, or ^Z, because that is how the system echoes control key sequences.

Unless otherwise noted, all numeric values are represented in decimal notation.

Unless otherwise specified, you terminate command lines by pressing the RETURN key.

SUMMARY OF TECHNICAL CHANGES

This manual documents VAX-11 DIGITAL Standard Runoff (DSR) Version 2.0 as released with Version 3.0 of VAX/VMS. This section summarizes the technical changes in the use of DSR from earlier versions.

COMMAND-LINE CHANGES

- New qualifiers, /NONSPACING_UNDERLINE and /SEPARATE_UNDERLINE, have been added. The syntax of the /UNDERLINE qualifier has been modified accordingly.
- New qualifiers, /LOG and /NOLOG, have been added to control whether DSR outputs a termination message.
- The VAX/VMS convention, /NOOUTPUT, is now supported in DSR.

NEW DSR COMMANDS

- .SET LEVEL
- .KEEP and .NO KEEP
- .AUTOJUSTIFY and .NO AUTOJUSTIFY
- .XLOWER and .XUPPER

CHANGES TO EXISTING DSR COMMANDS

- Six new arguments have been added to the .STYLE HEADERS command.
- The .REQUIRE command now has a default file type of .RNO.
- Header levels whose text is wider than the margins now wrap and justify within the margins.
- The .NO NUMBER command does not affect running page numbers now.
- The .PAGING command now causes a page break.
- The .SEND TOC command no longer interprets flags within the text being sent to the table of contents; thus, it is possible to send flags with the .SEND TOC command. For example, you can now send flag pairs for emphasis such as boldfacing or underlining.

- Notes (specified with the .NOTE command) that are encountered by DSR after .NO JUSTIFY are both filled and justified. Previously, they were filled but not justified.

  Notes that are encountered by DSR after .NO AUTOJUSTIFY and .NO FILL are filled but not justified. Previously, they were justified as well as filled.

- The TOC and TCX utilities now take default answers to all subsequent questions if you respond with CTRL/Z to any question in their input dialogs.

CHAPTER 1

INTRODUCTION TO DSR

DIGITAL Standard Runoff (DSR) is a text-formatting program that produces formatted documents from draft copy.

When you use DSR for automatic text processing, you go through three steps:

1. Use a text editor to create or edit a text file.

2. Use DSR to process your file and format the text.

3. Print the formatted document.

To specify formatting with DIGITAL Standard Runoff, you include commands in the text file where you want them to take effect. You can specify the size of pages, an uneven right margin or a justified right margin, the amount of spacing to appear between lines, and the appearance of lists. You can do many other formatting tasks with DSR commands as well.

In addition to commands, DSR flags (special characters) are available. You can insert flags in your text to specify boldfacing, underlining, capitalization, and other formatting details. See Chapter 3 for a full discussion of flags and flag control commands.

The output file that results from DSR processing is a formatted document. Neither the commands nor the flags appear in it.

The following list contains DSR-related terms and conventions that you will find helpful:

- **Filling**--Filling means that successive words from the input text are added to a line until the addition of another complete word would exceed the right margin.

- **Justification**--Justification means that spacing is added between words to expand the line exactly to the right margin, making the margin even. (For example, this text is justified.)

- **Section**--A section is a portion of text such as a chapter or appendix.

- **Running head**--A running head consists of one or two lines of header information, followed by two blank lines, at the top of the page. The first line (by default) is the chapter title and page number. The second line (by default) is the first-level header for the paragraph that begins the page.

- **Header numbering**--A header is numbered according to the following:

  1. the number of the chapter or appendix in which it appears
  2. the number of the lower-level subsection that it introduces

  For example, a first-level header that introduces the third subsection in Chapter 1 is numbered like this:

  1.3 SUBSECTION TITLE

  A second-level header that introduces the first subsection of the previous subsection is numbered like this:

  1.3.1 Subsection Title

  and so on. However, note that by using certain section formatting commands (see Section 1.4.3) you can change the decimal header numbers that DSR provides--to letters or Roman numerals, for example.

- **Page numbering**--In documents that are not divided into sections, pages are sequentially numbered in the upper right-hand corner on every page but the first. In documents that are divided into sections, the page numbers are prefixed with an appropriate section number (for example, 1-2, 1-3, 1-4, and so on). By using certain page formatting commands (see Section 1.4.1), you can change the page numbering format that DSR provides.

- **Layout**--The layout of a page is the arrangement of running head information and page numbers.

1.1 DSR COMMAND FORMAT

A DSR command consists of the following parts:

- A **control flag** (.), which introduces a DSR command. Begin a command in column 1 unless it follows other commands in a command string.

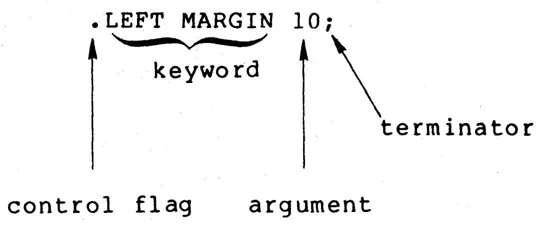
- A **keyword**, or its abbreviation, which immediately follows the control flag to specify the command function. A keyword is made up of a word or words separated by a space. The letters of a keyword may be entered in uppercase, lowercase, or mixed case.

- **Arguments**, which provide additional information for some commands. Use commas or spaces to separate multiple arguments (for example, .LAYOUT 1,3).

  Many commands have optional arguments. If you do not enter optional information, DSR supplies some predetermined standard numeric or alphabetic value (known as a default).

- A **terminator**, which ends the command or string of commands. Commands are most commonly terminated by the end of the line. Alternatively, you can terminate a command with a semicolon (;). You can terminate a command and begin a comment with an exclamation point (!). Or you can terminate a command and begin another one with a period (.).

Sample DSR command:



1.1.1 Abbreviating Commands

All commands have abbreviations. It is often convenient to enter the abbreviation instead of the full command. Legal abbreviations are given in the individual command descriptions in Chapter 2 and in the complete list of DSR command formats in Appendix C.

You can also type the beginning letters of the words forming the command keyword. You can use any such short form, as long as the truncation does not match any other command name or similarly shortened form of any other command name. For example, two of the possible short forms of the .NO CONTROL CHARACTERS command are these:

```
.NOCON CHA

.NO CO CH
```

Legal abbreviations (.NCC for this example) are preferred over the short forms since the latter cannot be guaranteed to be unique in a future version of DSR.

1.1.2 Entering Several Commands on a Line

You can put more than one command on a line if you follow these rules:

- You must type the first command in column 1 of a line.
- You can always put one command after another if all commands on the line either take no values or take numeric values.
- You can (except where explicitly disallowed) include a command that takes an alphabetic argument, as long as it is the last command on the line.
- You must precede each command with a control flag (.).

Example:

```
.BLANK.LEFT MARGIN 0.INDENT 10
```

There are exceptions to these rules. Some commands that take alphabetic values (such as the .DISPLAY commands) can appear anywhere in a line of commands. Other commands take text, but they must be followed by a semicolon (;) for another command to follow on the same line. Chapter 2 gives descriptions of the formats of the commands.

You terminate a command or line of commands by ending the line. However, if you want to keep text on the same line with a command, terminate the command by typing a semicolon instead. Text immediately follows the semicolon, with no space in between:

```
We sail the ocean blue,
.BLANK;And our saucy ship's a beauty.
```

In the above example, the semicolon after the .BLANK command tells DSR that the command is terminated and text now follows. Therefore, DSR inserts a blank line between the two lines of text.

1.1.3 Separating Command Arguments

There are rules for separating command arguments from keywords, and for separating arguments from other arguments. The rule for separating arguments from keywords is: If the leading character of the first argument is not a letter, no separator is required; but if the leading character is a letter, the letter must be separated from the final keyword by at least one space or tab. For example:

```
.TITLE Runoff
```

is acceptable, whereas

```
.TITLERunoff
```

is not, because DSR cannot differentiate between the argument and the keyword.

The rule for separating arguments from other arguments is: If more than one argument is required, you may insert a space- or comma-separator between arguments. But, if you have adjacent letters or numbers in your argument sequence, you must insert a separator. A space-separator consists of at least one SPACE or TAB character. A comma-separator consists of a single comma, alone or within any number of spaces and/or tabs.

Null arguments

If you wish to use a default value when you are entering a sequence of arguments, use a comma to indicate a null argument. DSR assigns the appropriate default value to the null argument.

In the following example, the .DISPLAY ELEMENTS command invokes default values for the first and second arguments ("q" and y) while using a given value ("x") for the third argument:

```
Format:   .DISPLAY ELEMENTS "q",y,"q"

Example:  .DISPLAY ELEMENTS ,,"x"
```

1.2 COMMAND DEFAULTS

Suppose you have not included any DSR commands at all in your text file. When you process this file with DSR, you will still see a difference in the way the text looks in the output file. The reason for the difference is that DSR provides certain basic formats just as if you had included the appropriate commands in your text file. These functions, known as command defaults, are as follows:

1. A standard typewriter page size of 8 1/2 x 11 inches; that is, a width of 60 character positions and a length of 58 lines of text per page (.PAGE SIZE 58,60)
2. Sequential page numbering for every page but the first (.PAGING)
3. A left margin setting of 0 (just before the first character position of a line) and a right margin setting of 60 (just after the 60th character position of a line) (.LEFT MARGIN 0 and .RIGHT MARGIN 60)
4. Line spacing equivalent to the single-space setting on a typewriter (.SPACING 1)
5. A tab setting every eighth character position on a line (.TAB STOPS 9,17, 25...)
6. Filling (.FILL)
7. Justification (.JUSTIFY)

1.3 DSR FORMATTING EXAMPLES

The following examples show how to use a few DSR commands and flags to format text. Complete descriptions of the commands are given in Chapter 2. Flags are described in Chapter 3.

1.3.1 Example 1: Formatting Text with DSR Defaults

Follow step 1 to create an input file named DEMO1.RNO, and then follow steps 2 and 3 to create and print the output file.

1. Create the input file, DEMO1.RNO, by using either a text editor or the CREATE command:

```
          "Annabel Lee"
It was many and many a year ago,
  In a kingdom by the sea,
That a maiden there lived whom you may know
  By the name of ANNABEL LEE;
And this maiden she lived with no other thought
  Than to love and be loved by me.
                -- Edgar Allan Poe
```

2. Create a DSR output file by using the RUNOFF command:

```
     $ RUNOFF DEMO1.RNO
```

   The input file (DEMO1.RNO) is now processed. DSR creates an output file with the file type .MEM (DEMO1.MEM).

   If DSR detects any error during processing, the type of error and its location in both the input and output files are printed. If an error occurs, return to step 1 to edit the file; otherwise, proceed to step 3.

3. Print the document:

```
$ PRINT DEMO1.MEM
```

The formatted output file is this:

```
"Annabel Lee" It was many and many a year ago, In a  kingdom
by  the  sea, That a maiden there lived whom you may know By
the name of ANNABEL LEE;  And this maiden she lived with  no
other  thought  Than  to  love and be loved by me.  -- Edgar
Allan Poe
```

Notice that the poetry format used for the input file has been ignored by DSR in producing the output file. Without specific DSR commands, only defaults are used to shape the text. Thus, the text of the output file has default margins, single-spacing, filling, and justification.

1.3.2 Example 2: Formatting Text with Some DSR Commands and Flags

Repeat step 1 to create an input file having the file specification DEMO2.RNO, and then use steps 2 and 3 to create and print the output file.

The input file is this:

```
.CENTER;^&Annabel Lee\&
.BLANK;It was many and many a year ago,
.INDENT 2;In a kingdom by the sea,
.BLANK;That a maiden there lived whom you may know
.INDENT 2;By the name of ANNABEL LEE;
.BLANK;And this maiden she lived with no other thought
.INDENT 2;Than to love and be loved by me.
.BLANK 2; TAB    TAB    TAB    Edgar Allan Poe
```

The formatted output file is this:

```
                  Annabel Lee
                  -------  ---

It was many and many a year ago,
  In a kingdom by the sea,

That a maiden there lived whom you may know
  By the name of ANNABEL LEE;

And this maiden she lived with no other thought
  Than to love and be loved by me.


                      -- Edgar Allan Poe
```

Here is a list of the DSR commands used in Example 2, with a brief explanation of each:

.CENTER	Centers the following text on the page width
.BLANK	Leaves one blank line
.INDENT 2	Indents the line two character positions to the right
.BLANK 2	Leaves two blank lines

In addition to the commands, there are two flags:

^&	Starts underlining the following text
\&	Stops underlining text

In the final version, notice the following:

- The .BLANK command has caused a blank line to occur before each line of verse
- The .INDENT 2 command has caused every other line of text to be indented two spaces
- The .CENTER command has caused the title to be centered on the line
- DSR flag characters have turned on underlining (^&) and turned it off (\&)

1.3.3 Example 3: Formatting a Term Paper

The following examples show how to use a few DSR commands to format a term paper, including a title page and a footnote.

Formatting a title page

Create an input file, called TITLE.RNO, that contains the following text and commands:

```
.FIGURE 20
.CENTER;THE AMERICAN REVOLUTIONARY ERA
.BLANK
.CENTER;POLITICAL SENTIMENT AS REFLECTED IN SONGS FROM 1759 TO 1800
.BLANK 2
.CENTER;by
.BLANK
.CENTER;Gilbert W. Schwenk
```

Process TITLE.RNO with DSR. Example 1-3 shows the output file.

Example 1-3: Title Page

THE AMERICAN REVOLUTIONARY ERA

POLITICAL SENTIMENT AS REFLECTED IN SONGS FROM 1759 TO 1800

by

Gilbert W. Schwenk

The .FIGURE command allows you to position the title in the middle of the page. Determine how far down from the top of the page you want the first line of text to appear, estimate how many lines of space that is, and use that number with the .FIGURE command to position the title.

Formatting text containing a footnote

Create an input file, called TEXT.RNO, that contains the following text and commands:

```
.SPACING 2
.INDENT 6;No doubt the best-known song of the American
Revolution is "Yankee Doodle."  It began as a British tune;
the British sang "Yankee Doodle" to make fun of the Yankees,
but the Yankees adopted it as their own song after they
routed the British troops from Concord.  The most familiar
words were written by Edward Bangs, a Minuteman who is said
to have been at Concord and Lexington.  He wrote the words
sometime after July 3, 1775--the date on which George
Washington took command of the American army in Boston.
.INDENT 6;Here is a representative stanza from Bangs'
version of "Yankee Doodle":
.SKIP.SPACING 1
.LEFT MARGIN 10
.NO FILL
Fath'r and I went down to camp
Along with Captain Gooding;
And there we saw the men and boys
As thick as hasty pudding.+
.FOOTNOTE
.SPACING 1
.REPEAT 15 "-"
.SKIP
.LEFT MARGIN 3.INDENT -3
+##Irwin Silber, ^&Songs of Justice\& (Harrisburg, PA:
Stackpole Books, 1973), p.36
.END FOOTNOTE
.SKIP.FILL.LEFT MARGIN 0
.SPACING 2
.INDENT 6;Naturally there were many variations of the song,
and many parodies. The Loyalists jumped in with their own
versions, such as "Adam's Fall; or, The Trip to Cambridge."
```

Process TEXT.RNO with DSR. Example 1-4 shows the output file.

Example 1-4: Text with Footnote

No doubt the best-known song of the American Revolution is "Yankee Doodle." It began as a British tune; the British sang "Yankee Doodle" to make fun of the Yankees, but the Yankees adopted it as their own song after they routed the British troops from Concord. The most familiar words were written by Edward Bangs, a Minuteman who is said to have been at Concord and Lexington. He wrote the words sometime after July 3, 1775--the date on which George Washington took command of the American army in Boston.

Here is a representative stanza from Bangs' version of "Yankee Doodle":

> Fath'r and I went down to camp
> Along with Captain Gooding;
> And there we saw the men and boys
> As thick as hasty pudding.+

Naturally there were many variations of the song, and many parodies. The Loyalists jumped in with their own versions, such as "Adam's Fall; or, The Trip to Cambridge."

+ Irwin Silber, Songs of Justice (Harrisburg, PA: Stackpole Books, 1973), p.36

Here is a list of the new commands and flags used in the text/footnote example, with a brief explanation of each:

.SPACING 2	Specifies double-spacing
.SPACING 1	Specifies single-spacing
.SKIP	Leaves a number of blank lines according to the .SPACING value
.NO FILL	Disables line-filling
.FILL	Enables line-filling
.FOOTNOTE	Indicates that the following text is to be put in a footnote
.END FOOTNOTE	Indicates the end of footnote text
.REPEAT 15 "-"	Repeats the hyphen 15 times
.INDENT -3	Causes the following text to be indented to the left

In the final version, notice the following:

- The .REPEAT 15 "-" command causes 15 hyphens to form a line that separates the footnote from the body of the text.
- The .NO FILL command causes DSR to suspend line-filling so that the four lines of the song remain in stanza form; without this command DSR would ignore your format of the four lines, and they would be run together to fill up one or two complete lines. The .FILL command resumes line-filling for the rest of the text.
- The .FOOTNOTE and .END FOOTNOTE commands cause a footnote to appear at the bottom of the page.
- The .INDENT -3 command causes the plus sign in the footnote to be placed at the left margin, while the text lines of the footnote are indented 3 spaces (because of the .LEFT MARGIN 3 command).

1.3.4 Example 4: Formatting a User's Guide

The following example shows how to use a few DSR commands to format a simple user's guide that includes header levels, a table, a list, and a table of contents (shown in a later section).

Formatting text containing header levels, a table, and a list

Create an input file, called GUIDE.RNO, that contains the following text and commands:

```
.CHAPTER EDT keypad editing with the VT100
.HEADER LEVEL 1 Introduction
The keypad is the set of keys to the right of the keyboard.
You will probably want to use keypad editing on the VT100
because it is
.BLANK.LIST
.LIST ELEMENT;Convenient--You can use most editing
functions by pressing one or two keypad keys.
.LIST ELEMENT;Fast--You press keys rather than type
commands, so you save time and avoid typographical errors.
.LIST ELEMENT;Versatile--You can customize EDT to meet
your needs by redefining keys.
.END LIST
.HEADER LEVEL 1 Keypad functions
Keypad keys are labeled with characters such as "ENTER" or
"5." Functions are assigned to each of the keypad keys.
Most of the keys have two functions.  You can use either of
the two functions, depending on whether or not you press the
GOLD key first.  (The GOLD key is the key labeled PF1 on the
keypad.)  To use the standard function, press the key.  To
use the alternate function, press GOLD first and then the
key.  Here is a table listing the standard and alternate
functions of each keypad key:
.BLANK.LEFT MARGIN 10
.CENTER;TABLE 1:  KEYPAD FUNCTIONS
.BLANK.TAB STOPS 18,40
^&Key TAB Standard Function TAB Alternate Function\&
.BLANK.NO FILL
.TAB STOPS 21,42
PF1 TAB GOLD TAB None
PF2 TAB HELP TAB None
PF3 TAB Find next TAB Find
PF4 TAB Delete Line TAB Undelete Line
#9 TAB Append TAB Replace
#8 TAB Section TAB Fill
#7 TAB Page TAB Command
#.
#.
#.
.BLANK.LEFT MARGIN 0.FILL
```

```
.HEADER LEVEL 1 Moving the cursor
The functions described in this section let you move the
cursor and set the direction of cursor movement during your
editing session.
.HEADER LEVEL 2 The Arrow Keys
You can move the cursor up, down, to the left, and to the
right with the four arrow keys.  The arrow keys are found on
the keyboard to the left of the keypad.
.HEADER LEVEL 2 Setting the Direction of Cursor Movement
ADVANCE and BACKUP change the direction in which the cursor
moves.  ADVANCE sets the cursor direction forward through
the file, that is, to the right and down.  BACKUP sets the
cursor direction backward through the file, that is, to the
left and up.
.HEADER LEVEL 2 Movement by Entity
You can move the cursor by entities of text.  An entity is a
character string that EDT recognizes as a unit, for example,
a word or paragraph.  The direction of cursor movement by
entity depends on whether EDT is set to ADVANCE or BACKUP.
.HEADER LEVEL 1 Deleting and inserting text
The functions described in this section let you delete and
insert text.
```

Process GUIDE.RNO with DSR. Example 1-5 shows the output file.

Example 1-5: User's Guide

CHAPTER 1

EDT KEYPAD EDITING WITH THE VT100

1.1 INTRODUCTION

The keypad is the set of keys to the right of the keyboard. You will probably want to use keypad editing on the VT100 because it is:

1. Convenient--You can use most editing functions by pressing one or two keypad keys.

2. Fast--You press keys rather than type commands, so you save time and avoid typographical errors.

3. Versatile--You can customize EDT to meet your needs by redefining keys.

1.2 KEYPAD FUNCTIONS

Keypad keys are labeled with characters such as "ENTER" or "5." Functions are assigned to each of the keypad keys. Most of the keys have two functions. You can use either of the two functions, depending on whether or not you press the GOLD key first. (The GOLD key is the key labeled PF1 on the keypad.) To use the standard function, press the key. To use the alternate function, press GOLD first and then the key. Here is a table listing the standard and alternate functions of each keypad key:

TABLE 1: KEYPAD FUNCTIONS

Key	Standard Function	Alternate Function
PF1	GOLD	None
PF2	HELP	None
PF3	Find Next	Find
PF4	Delete Line	Undelete Line
9	Append	Replace
8	Section	Fill
7	Page	Command
.		
.		
.		

1.3 MOVING THE CURSOR

The functions described in this section let you move the cursor and set the direction of cursor movement during your editing session.

1.3.1 The Arrow Keys

You can move the cursor up, down, to the left, and to the right with the four arrow keys. The arrow keys are found on the keyboard to the left of the keypad.

1.3.2 Setting The Direction Of Cursor Movement

ADVANCE and BACKUP change the direction in which the cursor moves. ADVANCE sets the cursor direction forward through the file, that is, to the right and down. BACKUP sets the cursor direction backward through the file, that is, to the left and up.

1.3.3 Movement By Entity

You can move the cursor by entities of text. An entity is a character string that EDT recognizes as a unit, for example, a word or a paragraph. The direction of cursor movement by entity depends on whether EDT is set to ADVANCE or BACKUP.

1.4 DELETING AND INSERTING TEXT

The functions described in this section let you delete and insert text.

Here is a list of new commands used in the user's guide example, with a brief explanation of each:

.CHAPTER	Starts and numbers a chapter
.HEADER LEVEL 1	Specifies a first-level header
.HEADER LEVEL 2	Specifies a second-level header
.LIST	Specifies the beginning of a list
.LIST ELEMENT	Specifies an item in a list
.END LIST	Specifies the end of a list
.TAB STOPS 18,40	Sets tab stops at the 18th and 40th character positions on a line
.TAB STOPS 21,42	Resets the tab stops to character positions 21 and 42

In the final version, notice the following:

- The .CHAPTER command numbers the chapter, puts the chapter title in uppercase, and centers the title 12 lines down from the top of the page.
- The .HEADER LEVEL commands number the header levels. First-level headers are put in uppercase, while second-level headers have only the first letters of each word in uppercase.
- The .LIST, .LIST ELEMENT, and .END LIST commands form and number a list of items.

Creating a Table of Contents

To create a table of contents in DSR, you use the TOC (Table of Contents) program. This example shows the steps you go through in creating a table of contents. For detailed information on TOC, see Chapter 4. (DSR qualifiers are described in Chapter 6.)

For this example, use the file GUIDE.RNO that you created in the previous example.

Step 1: Process the file, specifying the /CONTENTS qualifier after the RUNOFF command:

```
$ RUNOFF/CONTENTS GUIDE.RNO
```

DSR processes the file and creates a .MEM file as it normally does. It also creates a file called GUIDE.BTC. The .BTC file is the one that you process with TOC.

Step 2: Run TOC to process GUIDE.BTC. To run TOC, type this:

```
$ RUN SYS$SYSTEM:TOC
```

TOC then prompts you for the file name:

```
Specify input file:
GUIDE.BTC
```

Then TOC asks six questions about the format of the table of contents. For the purposes of this example, answer each question with the default answer (shown in parentheses) by pressing the RETURN key after each prompt. For example:

```
Varying header-level indents?  [Y/N] (N)
RET
```

When you have answered all the questions, TOC prints FINISHED and the dollar-sign prompt appears. TOC creates a file GUIDE.RNT. The .RNT file contains the table of contents plus some DSR commands.

Step 3: To get a finished table of contents, you must process the .RNT file with DSR:

```
$ RUNOFF GUIDE.RNT
```

DSR creates a file with the type .MEC. GUIDE.MEC is the finished table of contents and looks like Example 1-6.

Example 1-6: Table of Contents

```
CHAPTER 1       EDT KEYPAD EDITING WITH THE VT100

        1.1     INTRODUCTION . . . . . . . . . . . . . . . . . . . . . 1-1
        1.2     KEYPAD FUNCTIONS . . . . . . . . . . . . . . . . . . . 1-1
        1.3     MOVING THE CURSOR  . . . . . . . . . . . . . . . . . . 1-2
        1.3.1     The Arrow Keys . . . . . . . . . . . . . . . . . . . 1-2
        1.3.2     Setting the Direction of Cursor Movement . . . . . . 1-2
        1.3.3     Movement by Entity . . . . . . . . . . . . . . . . . 1-2
        1.4     DELETING AND INSERTING TEXT  . . . . . . . . . . . . . 1-2
```

1.4 WHAT THE DSR COMMANDS DO

This section contains a list of DSR commands grouped according to function. These commands are described in Chapter 2, where they are listed alphabetically. The flags and flag control commands are described in Chapter 3.

1.4.1 Page Formatting Commands

Page formatting commands allow you to control the following items:

- The size of the text area relative to the size of the paper on which it is printed
- The appearance and format of running heads
- The appearance and format of page numbering
- Optional subpaging

1.4.1.1 Page Size and Running Heads

1.4.1.1 Page Size and Running Heads - These commands allow you to control page size and running head formats:

.AUTOSUBTITLE and .NO AUTOSUBTITLE
.DATE and .NO DATE
.FIRST TITLE
.HEADERS ON and .NO HEADERS
.HEADERS UPPER, .HEADERS LOWER, and .HEADERS MIXED
.LAYOUT
.PAGE SIZE
.SUBTITLE and .NO SUBTITLE
.TITLE

1.4.1.2 Paging and Page-Number Contrcl - These commands allow you to control page numbering format:

.DISPLAY NUMBER
.NUMBER PAGE and .NO NUMBER
.NUMBER RUNNING
.PAGING and .NO PAGING

1.4.1.3 Subpaging - These commands allow you to control subpaging format:

.DISPLAY SUBPAGE
.NUMBER SUBPAGE
.SUBPAGE and .END SUBPAGE

1.4.2 Text Formatting Commands

Text formatting commands allow you to control the following items:

- Margin settings
- The amount of spacing between lines of text
- Filling and justification of text
- Indent and tab stop settings
- Paragraph formation
- The appearance of figures, lists, notes, and footnotes
- Text emphasis (for example, boldfacing or underlining)

1.4.2.1 Margin Setting - These commands allow you to control the margin settings:

.LEFT MARGIN
.RIGHT MARGIN

1.4.2.2 Filling and Justification

These commands allow you to control filling and justification:

.AUTOJUSTIFY and .NO AUTOJUSTIFY
.FILL and .NO FILL
.JUSTIFY and .NO JUSTIFY

1.4.2.3 Vertical Spacing

These commands allow you to control the amount of spacing between lines of text, and also paging:

.BLANK
.BREAK
.KEEP and .NO KEEP
.SKIP
.SPACING
.PAGE
.TEST PAGE

1.4.2.4 Horizontal Spacing

These commands allow you to control the spacing and positioning of text:

.CENTER
.INDENT
.PERIOD and .NO PERIOD
.RIGHT
.TAB STOPS

1.4.2.5 Paragraph Formatting

These commands allow you to control the format of paragraphs:

.AUTOPARAGRAPH and .NO AUTOPARAGRAPH
.AUTOTABLE and .NO AUTOTABLE
.PARAGRAPH
.SET PARAGRAPH

1.4.2.6 Text Emphasis

These commands allow you to control the appearance of text emphasis (such as boldfacing or underlining):

.ENABLE BAR, .DISABLE BAR, .BEGIN BAR, and .END BAR
.ENABLE BOLDING and .DISABLE BOLDING
.ENABLE HYPHENATION and .DISABLE HYPHENATION
.ENABLE OVERSTRIKING and .DISABLE OVERSTRIKING
.ENABLE UNDERLINING and .DISABLE UNDERLINING

1.4.2.7 Figures

These commands allow you to control the format of figures:

.FIGURE DEFERRED and .FIGURE
.LITERAL and .END LITERAL

1.4.2.8 Lists - These commands allow you to control the appearance and format of lists:

.DISPLAY ELEMENTS
.LIST and .END LIST
.LIST ELEMENT
.NUMBER LIST

1.4.2.9 Notes and Footnotes - These commands allow you insert notes and footnotes:

.FOOTNOTE and .END FOOTNOTE
.NOTE and .END NOTE

1.4.3 Section Formatting Commands

Section formatting commands allow you to divide your document into chapters or sections and to include tables of contents, appendixes, and indexes.

1.4.3.1 Appendixes and Chapters - These commands allow you to control the appearance and format of appendixes and chapters:

.APPENDIX
.CHAPTER
.DISPLAY APPENDIX
.DISPLAY CHAPTER
.NUMBER APPENDIX
.NUMBER CHAPTER

1.4.3.2 Sections - These commands allow you to control the appearance and format of sections:

.DISPLAY LEVELS
.HEADER LEVEL
.NUMBER LEVEL
.SET LEVEL
.STYLE HEADERS

1.4.3.3 Indexes - These commands allow you to control the appearance of indexes and index entries:

.DO INDEX
.ENABLE INDEXING and .DISABLE INDEXING
.ENTRY
.FLAGS INDEX and .NO FLAGS INDEX
.FLAGS SUBINDEX and .NO FLAGS SUBINDEX
.INDEX
.PRINT INDEX
.XLOWER and .XUPPER

1.4.3.4 Tables of Contents - These commands allow you to control the appearance of tables of contents and table-of-contents entries:

.ENABLE TOC and .DISABLE TOC
.SEND TOC

1.4.4 Flag Recognition Commands

Flag recognition commands enable or disable DSR's recognition of flag characters (see Chapter 3).

.FLAGS ACCEPT and .NO FLAGS ACCEPT
.FLAGS ALL and .NO FLAGS ALL
.FLAGS BOLD and .NO FLAGS BOLD
.FLAGS BREAK and .NO FLAGS BREAK
.FLAGS CAPITALIZE and .NO FLAGS CAPITALIZE
.FLAGS COMMENT and .NO FLAGS COMMENT
.FLAGS HYPHENATE and .NO FLAGS HYPHENATE
.FLAGS LOWERCASE and .NO FLAGS LOWERCASE
.FLAGS OVERSTRIKE and .NO FLAGS OVERSTRIKE
.FLAGS PERIOD and .NO FLAGS PERIOD
.FLAGS SPACE and .NO FLAGS SPACE
.FLAGS SUBSTITUTE and .NO FLAGS SUBSTITUTE
.FLAGS UNDERLINE and .NO FLAGS UNDERLINE
.FLAGS UPPERCASE and .NO FLAGS UPPERCASE

1.4.5 Miscellaneous Commands

These commands allow you to do various things such as inserting a date and time, repeating characters, and including other files.

.CONTROL CHARACTERS and .NO CONTROL CHARACTERS
.IF, .IFNOT, .ELSE, and .ENDIF
.NO SPACE
.REPEAT
.REQUIRE
.SET DATE and .SET TIME
.VARIABLE

CHAPTER 2

THE DSR COMMANDS

This chapter contains an alphabetic list of all DSR commands (except for the flag control commands, which are described in Chapter 3). Under each command are descriptions, as applicable, of the command's function, format, characteristics, and defaults, as well as the effects other commands may have on the command. Under Characteristics you will find any side effects of the command, other commands that may be required, and other commands that might be issued by that command.

Legal abbreviations for the commands are included in the Format sections.

.APPENDIX

.APPENDIX

Function:

The .APPENDIX command specifies the beginning of an appendix, assigns an identifying letter to it, and allows you to supply a title. Successive .APPENDIX commands assign identifying letters in alphabetical order. (See also .NUMBER APPENDIX and .DISPLAY APPENDIX.)

Format:

```
.APPENDIX [text]

.AX [text]
```

text

The title you give the appendix

Characteristics:

- .APPENDIX issues a .BREAK before doing its main task.
- .APPENDIX issues a .LEFT MARGIN 0 and a .SPACING 1.
- .APPENDIX issues a .FILL. It also issues a .JUSTIFY unless you have issued a .NO AUTOJUSTIFY. (However, note that if .JUSTIFY was in effect before you issued the .APPENDIX, .NO AUTOJUSTIFY does not cancel .JUSTIFY.)
- .APPENDIX issues a .PAGING command.
- .APPENDIX sets the right margin to the current page width (see .PAGE SIZE).
- .APPENDIX issues a .PAGE and inserts 12 blank lines.
- .APPENDIX prints and centers the word appendix, following it with a space and a letter identifying the appendix. After printing a blank line, .APPENDIX prints the title in uppercase unless you have specified otherwise with case flags (see Chapter 3). Three blank lines follow the title.
- If you issue an .APPENDIX command after or instead of a .TITLE command, the appendix title becomes the running head title and any .SUBTITLE in effect before the .APPENDIX is canceled.
- .APPENDIX is treated as a .CHAPTER command if you issue it immediately after a .NUMBER CHAPTER command.

.AUTOJUSTIFY and .NO AUTOJUSTIFY

Function:

When you issue .AUTOJUSTIFY, the following commands automatically issue .JUSTIFY (as well as .FILL) commands:

.APPENDIX
.CHAPTER
.HEADER LEVEL
.NOTE

If you disable automatic justification by issuing .NO AUTOJUSTIFY, DSR does not disturb the justify/no-justify state that is in effect (whether by default, or as a result of a previous .JUSTIFY or .NO JUSTIFY command) at the time you use one of these commands. Whichever state is in effect remains in effect when you issue .NO AUTOJUSTIFY. (See also .JUSTIFY, .NO JUSTIFY, .FILL, and .NO FILL.)

Format:

Enable	Disable
.AUTOJUSTIFY	.NO AUTOJUSTIFY
.AJ	.NAJ

Default:

.AUTOJUSTIFY

.AUTOPARAGRAPH

.AUTOPARAGRAPH and .NO AUTOPARAGRAPH

Function:

The .AUTOPARAGRAPH and .NO AUTOPARAGRAPH commands turn the automatic paragraph capability on and off. If .AUTOPARAGRAPH is in effect and you start a line with a space or tab, DSR automatically formats a new paragraph. It is formatted according to .PARAGRAPH or .SET PARAGRAPH values, whether they are specified or supplied by default (see .PARAGRAPH). .AUTOPARAGRAPH functions the same way that .AUTOTABLE does, except that .AUTOTABLE starts a new paragraph each time a line does NOT start with a space or tab (see .AUTOTABLE).

Format:

Enable	Disable
.AUTOPARAGRAPH	.NO AUTOPARAGRAPH
.AP	.NAP

Characteristics:

- .FILL must be in effect for a space or tab to start a new paragraph.
- If you issue either .AUTOPARAGRAPH or .NO AUTOPARAGRAPH, you cancel .AUTOTABLE.

Effects of other commands:

- If you issue .AUTOTABLE or .NO AUTOTABLE, you cancel .AUTOPARAGRAPH.

Defaults:

If you have not issued a .PARAGRAPH or .SET PARAGRAPH, DSR executes a .TEST PAGE 2 followed by a .SKIP 1 and a .INDENT 5.

Example:

Example 2-1 illustrates the use of the .AUTOPARAGRAPH command.

Example 2-1: .AUTOPARAGRAPH

The following shows the input text format before it is processed by DSR:

```
.AUTOPARAGRAPH
-------------------------------------------------
---------------------------------------------
-----------------------------------------------
-------------------------------------------------
(TAB)-----------------------------------------------
----------------------------------------------
------------------------------------------------
```

The output text format looks like this:

```
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------

     ---------------------------------------------
--------------------------------------------------
--------------------------------------------------
```

.AUTOSUBTITLE

.AUTOSUBTITLE and .NO AUTOSUBTITLE

Function:

The .AUTOSUBTITLE command causes DSR to use .HEADER LEVEL titles for running head subtitles. Subtitles are therefore able to change according to what section title applies to a given page. The .NO AUTOSUBTITLE command cancels the .AUTOSUBTITLE function. (See .HEADERS ON, .SUBTITLE, and .HEADER LEVEL.)

Format:

```
    Enable                         Disable

                  ┌+n┐
.AUTOSUBTITLE     │ n│         .NO AUTOSUBTITLE
                  └-n┘

     ┌+n┐
.AST │ n│                      .NAST
     └-n┘
```

n

The highest numbered .HEADER LEVEL whose title will be used as a subtitle. For example, if you issue .AUTOSUBTITLE 2, the titles of header levels 1 and 2 also appear as running head subtitles. Header levels 3 and above do not appear as running head subtitles.

+n

Increases the current highest numbered header level by n.

-n

Decreases the current highest numbered header level by n.

Characteristics:

- You must issue a .SUBTITLE command for .AUTOSUBTITLE to work.
- If the text of the header level that is used as a subtitle is wider than the margins currently in effect, the subtitle is truncated and an ellipsis (...) is appended to it.

Effects of other commands:

- The .DATE command causes the current date to be placed to the right of each subtitle.

Defaults:

- If you do not issue .AUTOSUBTITLE or .NO AUTOSUBTITLE, the default is .AUTOSUBTITLE 1.
- If you issue .AUTOSUBTITLE with no value, the default is the value you specified with the previous .AUTOSUBTITLE or 1 if no .AUTOSUBTITLE was previously issued.

.AUTOTABLE

.AUTOTABLE and .NO AUTOTABLE

Function:

The .AUTOTABLE and .NO AUTOTABLE commands turn the automatic paragraph capability on and off. If .AUTOTABLE is in effect, DSR formats a new paragraph for each line that does not start with a space or tab. It is formatted according to .PARAGRAPH or .SET PARAGRAPH values, whether they are specified or supplied by default (see .PARAGRAPH). .AUTOTABLE functions the same way that .AUTOPARAGRAPH does, except that .AUTOPARAGRAPH starts a new paragraph for each line that DOES start with a space or tab (see .AUTOPARAGRAPH).

Format:

Enable	Disable
.AUTOTABLE	.NO AUTOTABLE
.AT	.NAT

Characteristics:

- .FILL must be in effect for a new paragraph to start when you use .AUTOTABLE.
- If you issue either .AUTOTABLE or .NO AUTOTABLE, you cancel .AUTOPARAGRAPH.

Effects of other commands:

- If you issue .AUTOPARAGRAPH or .NO AUTOPARAGRAPH, you cancel .AUTOTABLE.

Defaults:

If you have not issued a .PARAGRAPH or .SET PARAGRAPH, the default is a .TEST PAGE 2 followed by a .SKIP 1 and an .INDENT 5.

Example:

Example 2-2 illustrates the use of the .AUTOTABLE command.

Example 2-2: .AUTOTABLE

The following shows the input text format before it is processed
by DSR:

```
.AUTOTABLE
-----------------------------------------------------
(TAB)------------------------------------------------------
(SP) ---------------------------------------------------------------
---------------------------------------------------------------
(SP) ----------------------------------------------------
---------------------------------------------------------
(TAB)-------------------------------------------------------
(TAB)-----------------------------------------------------
```

The output text format looks like this:

```
      ---------------------------------------------------------
---------------------------------------------------------------
---------------------------------------------------------------

      ---------------------------------------------------------
---------------------------------------------------------------

      ---------------------------------------------------------
---------------------------------------------------------------
---------------------------------------------------------------
```

.BLANK

.BLANK

Function:

The .BLANK command inserts exactly the number of blank lines that you specify. It is different from .SKIP, which inserts a multiple of the number of blank lines specified in the .SPACING command (see .SKIP and .SPACING).

Format:

$.BLANK \begin{bmatrix} -n \\ n \end{bmatrix}$

$.B \begin{bmatrix} -n \\ n \end{bmatrix}$

n

The number of blank lines you want inserted.

-n

Specifies that the next line will begin exactly n lines from the bottom of the current page.

Characteristics:

- The .BLANK command issues a .BREAK before doing its main task.
- .BLANK n does not work at the top of a page, that is, immediately following a .PAGE or just after .PAGE SIZE length has been exceeded. However, .BLANK -n pushes the next line to the bottom of the page (less n lines) under such conditions. (Use a .FIGURE command to insert blank lines at the top of a page.)
- If there is not enough room on the current page for .BLANK to do exactly as you specified, .BLANK does as much as it can on that page. It does not finish on the next page unless you issue .BLANK -n and .SPACING has a value greater than n.
- If DSR encounters a footnote while executing .BLANK, it considers the line directly above the footnote to be the bottom of the page.

Default:

If you issue .BLANK without a value, you get .BLANK 1.

.BREAK

Function:

The .BREAK command ends the current line immediately, without filling or justification. Issue a .BREAK when .FILL is in effect and you want a few short lines of text with no blank lines in between.

Format:

```
.BREAK

.BR

. RET
```

Characteristics:

- A .BREAK immediately after a .PARAGRAPH, .INDENT, .LEFT, .AUTOPARAGRAPH, or .AUTOTABLE cancels the indentation you just requested. This also occurs with most of the commands that issue a .BREAK command automatically.

Effects of other commands:

- The following DSR commands issue .BREAK commands before doing their main tasks:

```
.APPENDIX              .LIST and .END LIST
.BLANK                 .LIST ELEMENT
.CENTER                .PAGE
.CHAPTER               .PAGE SIZE
.DISPLAY APPENDIX      .PAPER SIZE
.DISPLAY CHAPTER       .PARAGRAPH
.DISPLAY ELEMENTS      .PRINT INDEX
.DISPLAY LEVELS        .RIGHT
.DISPLAY NUMBER        .RIGHT MARGIN
.DISPLAY SUBPAGE       .SET DATE
.DO INDEX              .SKIP
.FIGURE                .SPACING
.FIGURE DEFERRED       .STYLE HEADERS
.FILL and .NO FILL     .SUBPAGE and .END SUBPAGE
.HEADER LEVEL          .TEST PAGE
.INDENT                .TITLE
.LAYOUT
```

.CENTER

.CENTER (.CENTRE)

Function:

The .CENTER command centers a single line of text around a character position on a line (compare with .RIGHT).

Format:

```
        ┌+n┐                    ┌+n┐
.CENTER │ n│;text       .CENTER │ n│
        └-n┘            text    └-n┘

         or                 or

        ┌+n┐                    ┌+n┐
.CENTRE │ n│;text       .CENTRE │ n│
        └-n┘            text    └-n┘

  ┌+n┐                      ┌+n┐
.C│ n│;text           .C    │ n│
  └-n┘                 text└-n┘
```

n

Twice the value of the character position that you want to center the text around. (Absolute character positions on a line always start with 0 at the leftmost position on the page.)

If you center the line of text between settings of the left and right margins, then n is equal to the .LEFT MARGIN value added to the .RIGHT MARGIN value.

+n

Adds a value to the current value of n and normally is used to adapt .CENTER to a .LEFT MARGIN setting.

-n

Subtracts a value from the current value of n and normally is used to adapt .CENTER to a .RIGHT MARGIN setting.

text

The text you want centered. You must enter this text all on the same line.

Characteristics:

- The .CENTER command issues a .BREAK before doing its main task.
- The line of text being centered can extend past margin settings and even beyond the .PAGE SIZE width setting, but it cannot go to the left of character position 0.

- You can enter the text to be centered on the same line following the .CENTER command. Or you can end the line after the .CENTER command, in which case the following line of text is centered.

- No commands will be recognized on the line following a .CENTER command (or, if that line is blank, on the next line). The control flag (.) is not honored while .CENTER is collecting text to center. Other DSR flags are recognized, however, for example, the bold and underline flags.

- If the text being centered contains a semicolon (;), you must precede it with the Accept flag (_). Otherwise, DSR stops collecting text to center at the semicolon, issues an error message, and discards the rest of the line.

Default:

If you issue .CENTER without specifying n, the text is centered on the current page width.

.CHAPTER

.CHAPTER

Function:

The .CHAPTER command specifies the beginning of a chapter, numbers it, and allows you to supply a title. Successive .CHAPTER commands cause chapters to be sequentially numbered. (See also .NUMBER CHAPTER and .DISPLAY CHAPTER.)

Format:

```
.CHAPTER [text]

.CH [text]
```

text

The title of the chapter.

Characteristics:

- .CHAPTER issues a .BREAK before doing its main task.
- .CHAPTER issues a .LEFT MARGIN 0 and a .SPACING 1.
- .CHAPTER issues a .FILL. It also issues a .JUSTIFY, unless you have issued a .NO AUTOJUSTIFY. (However, note that if .JUSTIFY was in effect before you issued the .CHAPTER, .NO AUTOJUSTIFY does not cancel the .JUSTIFY.)
- .CHAPTER issues a .PAGING command.
- .CHAPTER sets the right margin to the current page width (see .PAGE SIZE).
- .CHAPTER issues a .PAGE and inserts 12 blank lines.
- .CHAPTER prints and centers the word <u>CHAPTER</u> and a number identifying the chapter. After printing a blank line, .CHAPTER prints the title in uppercase unless you have specified otherwise with case flags (see Chapter 3). Three blank lines follow the title.
- If you include a .CHAPTER after or instead of a .TITLE, the chapter title becomes the running head title and any .SUBTITLE in effect before the .CHAPTER is canceled.
- .CHAPTER is treated as an .APPENDIX command if you issue it immediately after a .NUMBER APPENDIX.

.CONTROL CHARACTERS and .NO CONTROL CHARACTERS

Function:

The .CONTROL CHARACTERS command causes DSR to accept control characters as normal text in your input file. (Control characters are characters whose ASCII representations are less than 40 [octal].) The .NO CONTROL CHARACTERS command cancels this capability.

Format:

Enable	Disable
.CONTROL CHARACTERS	.NO CONTROL CHARACTERS
.CC	.NCC

Default:

.NO CONTROL CHARACTERS

Example:

Example 2-3 shows how you might use the .CONTROL CHARACTERS and .NO CONTROL CHARACTERS commands in a command file.

Example 2-3: .CONTROL CHARACTERS

```
.!
.IF DIABLO
.! Switch the Diablo printer to horizontal 12-pitch (12 char/in.)
.! and vertical 7-pitch (7 char/in.). The codes are as follows:
.!
.!                   Codes          Decimal values
.! Horizontal:    <ESC><US><VT>     27 - 31 - 11
.! Vertical:      <ESC><RS><BS>     27 - 30 - 08
.!
.CONTROL CHARACTERS
<ESC>^_<VT>
<ESC>^^^H
.NO CONTROL CHARACTERS
.ENDIF DIABLO
.!
```

This command file allows device-dependent information to pass through DSR to the device (a DIABLO terminal). Without the .CONTROL CHARACTERS command DSR would send error messages and then delete the control characters.

.DATE

.DATE and .NO DATE

Function:

The .DATE and .NO DATE commands determine whether or not the current date appears in running heads. The date appears in the format dd mmm yy, for example, 22 Aug 81. A .SUBTITLE command must be included for .DATE to be effective. (See also .HEADERS ON and .SET DATE.)

Format:

Enable	Disable
.DATE	.NO DATE
.D	.ND

Characteristics:

- The date appears on the right-hand side of the subtitle line.

Effects of other commands:

- .DATE is not effective if either .LAYOUT 1 or .LAYOUT 2 is in effect.
- You can specify a different date with the .SET DATE command.

Default:

.NO DATE

.DISPLAY APPENDIX

Function:

The .DISPLAY APPENDIX command allows you to specify the form that sequential lettering (or numbering) of appendixes will take, whether in the title, the page numbers, or the first character of a header level number. This command does not change any values. (See also .APPENDIX and .NUMBER APPENDIX.)

Format:

```
.DISPLAY APPENDIX y

.DAX y
```

y

One of the following one- or two-letter codes:

Code	**Form of Sequence and Case**
D	Decimal Numbers
O	Octal Numbers
H	Hexadecimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
RM	Roman Mixed Case Numerals (only first numeral is uppercase)
LU	Letters, Uppercase
LL	Letters, Lowercase
LM	Letters, Mixed Case (only first letter is uppercase)

Characteristics:

- The .DISPLAY APPENDIX command issues a .BREAK before doing its main task.
- You issue a .DISPLAY APPENDIX before the .APPENDIX command you want to affect.

Default:

Uppercase letters (LU)

.DISPLAY CHAPTER

.DISPLAY CHAPTER

Function:

The .DISPLAY CHAPTER command allows you to specify the form that sequential numbering (or lettering) of chapters will take, whether in the title, the page numbers, or the first character of a header level number. This command does not change any values. (See also .CHAPTER and .NUMBER CHAPTER.)

Format:

```
.DISPLAY CHAPTER y

.DCH y
```

y

One of the following one- or two-letter codes:

Code	Form of Sequence and Case
D	Decimal Numbers
O	Octal Numbers
H	Hexadecimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
RM	Roman Mixed Case Numerals (only first numeral is uppercase)
LU	Letters, Uppercase
LL	Letters, Lowercase
LM	Letters, Mixed Case (only first letter is uppercase)

Characteristics:

- The .DISPLAY CHAPTER command issues a .BREAK before doing its main task.
- You issue a .DISPLAY CHAPTER before the .CHAPTER command you want to affect.

Default:

Decimal numbers (D)

.DISPLAY ELEMENTS

Function:

The .DISPLAY ELEMENTS command allows you to specify the form that sequential numbering or lettering of items in a list will take. This command does not change any values. (See also .LIST, .END LIST, and .NUMBER LIST.)

Format:

```
.DISPLAY ELEMENTS ["x",]y[,"z"] (or ['x',]y[,'z'])

.DLE ["x",]y[,"z"] (or ['x',]y[,'z'])
```

x

A character, such as a left parenthesis or bracket, that you can specify to precede the number or letter. You must enclose the character within quotation marks (" ") or apostrophes (' ').

y

One of the following one- or two-letter codes:

Code	Form of Sequence and Case
D	Decimal Numbers
O	Octal Numbers
H	Hexadecimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
RM	Roman Mixed Case Numerals (only first numeral is uppercase)
LU	Letters, Uppercase
LL	Letters, Lowercase
LM	Letters, Mixed Case (only first letter is uppercase)

z

A character, such as a right parenthesis or bracket, that you can specify to follow the number or letter. You must enclose the character within quotation marks (" ") or apostrophes (' ').

Characteristics:

- The .DISPLAY ELEMENTS command issues a .BREAK before doing its main task.

- You must issue a .DISPLAY ELEMENTS before the first .LIST ELEMENT command that you want to affect, but after the .LIST command.

- A .DISPLAY ELEMENTS command remains in effect only for a particular list. A list is defined by a .LIST command and its paired .END LIST. Other lists, similarly defined, can exist within it and are unaffected by the .DISPLAY ELEMENTS command issued for outer lists (or any other lists).

- If you omit a value from .DISPLAY ELEMENTS, the current setting remains unchanged, but you must retain any comma that would have followed it. (The final value present, however, need not have a comma after it.)

Defaults:

A space for x, decimal numbers for y, and a period (.) for z

.DISPLAY LEVELS

Function:

The .DISPLAY LEVELS command allows you to specify the form that sequential numbering (or lettering) of section headers will take. You can control the form of individual numbers within a section number for a header (that is, those numbers preceding or following a dot). This command does not change any values. (See also .HEADER LEVEL, .NUMBER LEVEL, and .STYLE HEADERS.)

Header Levels for Documents without Chapters and with Chapters

	Nonchapter Sections	Chapter n Sections
.HEADER LEVEL 1	1.0	n.1
.HEADER LEVEL 2	1.1	n.1.1
.HEADER LEVEL 3	1.1.1	n.1.1.1

	Index Sections	Appendix Sections
.HEADER LEVEL 1	INDEX.1	A.1
.HEADER LEVEL 2	INDEX.1.1	A.1.1
.HEADER LEVEL 3	INDEX.1.1.1	A.1.1.1

Format:

.DISPLAY LEVELS [y_1],[y_2],...[y_6]

.DHL [y_1],[y_2],...[y_6]

y

One of the following one- or two-letter codes; 1,2,...6 indicate positions of numbers (or letters) for a section header. The commas correspond to the dots in a printed section number. See the example under Characteristics. (See also .NUMBER LEVEL.)

Code	Form of Sequence and Case
D	Decimal Numbers
O	Octal Numbers
H	Hexadecimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
RM	Roman Mixed Case Numerals (only first numeral is uppercase)
LU	Letters, Uppercase
LL	Letters, Lowercase
LM	Letters, Mixed Case (only first letter is uppercase)

Characteristics:

- The .DISPLAY LEVELS command issues a .BREAK before doing its main task.

- If you have issued a .DISPLAY LEVELS RU,,LL command, for example, and you now issue a .HEADER LEVEL command that normally would produce a section number of 2.2.1, the section header number would appear as follows:

  1. For a document with no chapters --

     .HEADER LEVEL 3 produces II.2.a.

  2. For a document with chapters --

     .HEADER LEVEL 3, if you issued it in Chapter 1, produces 1.II.2.a. Note that this command does not affect the chapter number. However, .CHAPTER, .NUMBER CHAPTER, and .DISPLAY CHAPTER do affect it.

o y_1,y_2,...y_6 are displaced one position to the right if you have issued .CHAPTER, .APPENDIX, or .DO INDEX.

Default:

Decimal numbers (D)

.DISPLAY NUMBER

Function:

The .DISPLAY NUMBER command allows you to specify the form that sequential numbering (or lettering) of pages will take. This command does not change any values. (See also .HEADERS ON, .NUMBER PAGE, .NO NUMBER, .LAYOUT, .NUMBER RUNNING, and .NO PAGING.)

Format:

```
.DISPLAY NUMBER y

.DNM y
```

y

One of the following one- or two-letter codes:

Code	Form of Sequence and Case
D	Decimal Numbers
O	Octal Numbers
H	Hexadecimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
RM	Roman Mixed Case Numerals (only first numeral is uppercase)
LU	Letters, Uppercase
LL	Letters, Lowercase
LM	Letters, Mixed Case (only first letter is uppercase)

Characteristics:

- The .DISPLAY NUMBER command issues a .BREAK before doing its main task.
- You issue a .DISPLAY NUMBER before the page you want to affect. However, note that if you are using .LAYOUT 1 or .LAYOUT 2 (where the page number appears at the bottom of the page), the .DISPLAY NUMBER command might affect the page on which you issue that command when you are trying to affect the NEXT page.

Default:

Decimal numbers (D)

.DISPLAY SUBPAGE

.DISPLAY SUBPAGE

Function:

The .DISPLAY SUBPAGE command allows you to specify the form that sequential lettering (or numbering) of subpage characters will take. These are the characters that are appended to the page numbers of subpages. This command does not change any values. (See also .SUBPAGE and .NUMBER SUBPAGE.)

Format:

```
.DISPLAY SUBPAGE y

.DSP y
```

y

One of the following one- or two-letter codes:

Code	Form of Sequence and Case
D	Decimal Numbers
O	Octal Numbers
H	Hexadecimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
RM	Roman Mixed Case Numerals (only first numeral is uppercase)
LU	Letters, Uppercase
LL	Letters, Lowercase
LM	Letters, Mixed Case (only first letter is uppercase)

Characteristics:

- The .DISPLAY SUBPAGE command issues a .BREAK before doing its main task.
- You issue a .DISPLAY SUBPAGE before the subpage you want to affect.

Default:

Uppercase letters (LU) appended to the last page number

.DO INDEX

Function:

The .DO INDEX command prints an index containing alphabetized entries in a single-column format and starts the index on a new page. See Chapter 5, Creating an Index, for a more detailed discussion of this command.

Format:

```
.DO INDEX [text]

.DX [text]
```

.ENABLE BAR
.DISABLE BAR
.BEGIN BAR
.END BAR

.ENABLE BAR, .DISABLE BAR, .BEGIN BAR, and .END BAR

Function:

The bar commands control the insertion of vertical bars (|) at the beginning of lines of text. The bars (usually called change bars) are normally inserted to indicate where changes in text have occurred since the previous edition of a document. (You can specify a character other than vertical bars to indicate changes. See the description of the /CHANGE_BAR qualifier in Chapter 6, Running DSR.)

.ENABLE BAR shifts all text following it three characters to the right to make room for the bars on the left. The width of the lines of actual text is not altered.

.BEGIN BAR causes DSR to start inserting vertical bars at the beginning of lines.

.END BAR causes DSR to stop putting vertical bars at the beginning of lines.

.DISABLE BAR disables the bar commands but does not shift the lines of text back to their normal position.

Format:

```
.ENABLE BAR    .BEGIN BAR     .END BAR  .DISABLE BAR

.EBB           .BB            .EB       .DBB
```

Default:

No change bars and text not indented

Example:

Example 2-4 shows how to use the bar commands.

Example 2-4: BAR COMMANDS

The following shows the input file before it is processed by DSR:

```
.RIGHT MARGIN 45
A .BEGIN BAR command follows this text. The word
"Here" is barred. We should not see change bars
unless this file is processed with the
/CHANGE_BAR qualifier because we have not yet
issued an .ENABLE BAR command.
.BLANK
The next word is barred:
.BEGIN BAR
Here.
.END BAR
Did the change bar appear?
.BLANK
Following this line, we issue an .ENABLE BAR
command.
.ENABLE BAR
.BLANK
This text is unbarred.
.BLANK
.BEGIN BAR
But this text is barred.
.END BAR
.BLANK
Following this line, we turn off change bars again
with a .DISABLE BAR command.
.DISABLE BAR
.BLANK
The next word is barred:
.BEGIN BAR
Here.
.END BAR
Did the change bar appear?
.BLANK
Even though it did not, notice that the left margin
does not revert. It was offset 3 spaces to the
right when bars were enabled (either from the
command line or at the first .ENABLE BAR command).
```

When the example input file is processed through DSR without the /CHANGE_BAR qualifier, the output file is this:

```
A .BEGIN BAR command follows this text.  The  word
"Here"  is  barred.  We should not see change bars
unless this file is processed with the /CHANGE_BAR
qualifier  because  we  have  not  yet  issued  an
.ENABLE BAR command.
```

The next word is barred: Here. Did the change bar appear?

Following this line, we issue an .ENABLE BAR command.

This text is unbarred.

| But this text is barred.

Following this line, we turn off change bars again with a .DISABLE BAR command.

The next word is barred: Here. Did the change bar appear?

Even though it did not, notice that the left margin does not revert. It was offset 3 spaces to the right when bars were enabled (either from the command line or at the first .ENABLE BAR command).

.ENABLE INDEXING and .DISABLE INDEXING

Function:

These commands enable and disable, respectively, the operation of the indexing commands, the Index flag, and the /INDEX qualifier. See Chapter 5, Creating an Index, for a more detailed discussion.

Format:

Enable	Disable
.ENABLE INDEXING	.DISABLE INDEXING
.EIX	.DIX

Default:

```
.ENABLE INDEXING
```

.ENABLE TOC

.ENABLE TOC and .DISABLE TOC

Function:

These commands enable and disable DSR's collection of information for the table of contents. See Chapter 4, Creating a Table of Contents, for a more detailed discussion.

Format:

Enable	Disable
.ENABLE TOC	.DISABLE TOC
.ETC	.DTC

Default:

```
.ENABLE TOC
```

.ENTRY

Function:

The .ENTRY command creates an index entry without a page number reference. It is usually used for "see..." or "see also..." index entries. See Chapter 5, Creating an Index, for a more detailed discussion of this command.

Format:

.ENTRY topic[>subtopic$_1$...>subtopic$_n$]

.Y topic[>subtopic$_1$...>subtopic$_n$]

.FIGURE DEFERRED and .FIGURE

Function:

The .FIGURE DEFERRED command leaves room on a page for you to insert a figure later on. You specify the number of blank lines you need and DSR leaves that amount of space on the current page if there is enough room.

If there is not enough room on the current page, .FIGURE DEFERRED first adds enough text to complete the page and then puts the required number of blank lines at the top of the next page.

The .FIGURE command is the same as .FIGURE DEFERRED, except that if there is not enough room on the current page, DSR ends the page immediately and then puts the blank lines at the top of the next page.

Format:

```
    .FIGURE DEFERRED [n]      .FIGURE [n]

    .FGD [n]                  .FG [n]
```

n

The number of blank lines needed. Values of 0 or less are illegal and n cannot exceed the page length value associated with .PAGE SIZE.

Characteristics:

- The .FIGURE DEFERRED and .FIGURE commands both issue .BREAK commands before doing their main tasks.
- .FIGURE DEFERRED avoids short pages (a large amount of white space at the bottom of the page).
- You can issue .FIGURE and .PAGE alternately to produce consecutive blank pages. (A series of .PAGE commands alone does not accomplish this action. See also .SKIP and .BLANK.)
- For a figure ending a page, you can cause a caption to be printed at the bottom of the page by issuing the following commands after .FIGURE DEFERRED or .FIGURE:

```
.SKIP -1
.CENTER;figure caption
```

Defaults:

```
.FIGURE 1
.FIGURE DEFERRED 1
```

Example:

Example 2-5 shows how to use .FIGURE and .FIGURE DEFERRED.

Example 2-5: .FIGURE and .FIGURE DEFERRED

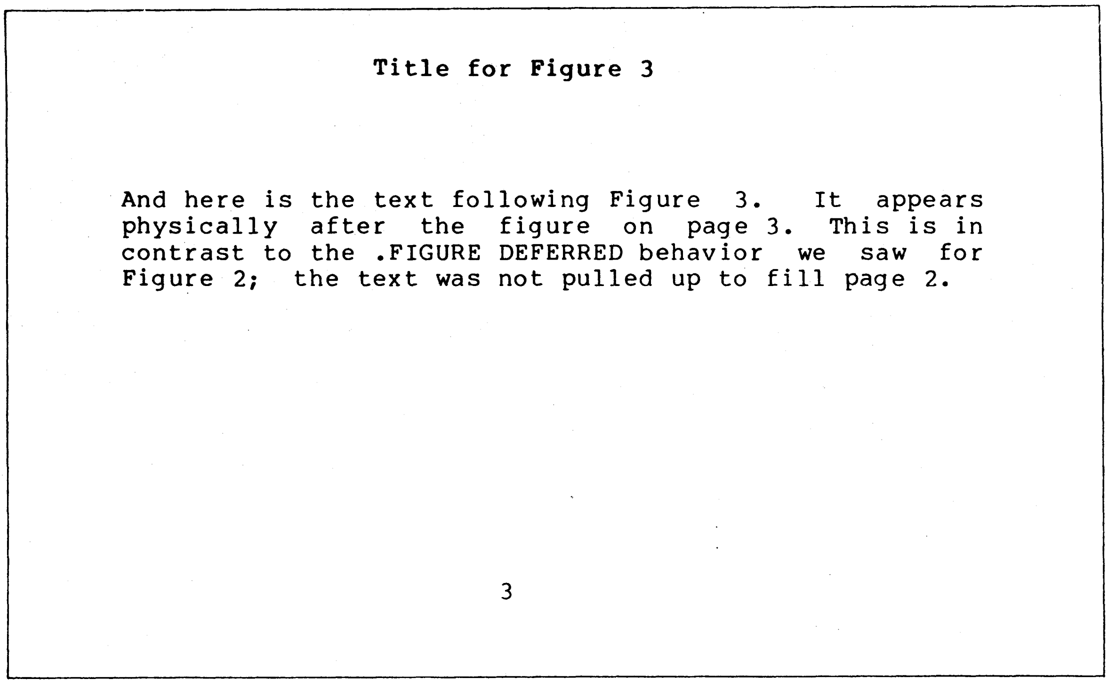
The following shows the input file before it is processed by DSR:

```
.PAGE SIZE 25,55
.LAYOUT 1,2
.FLAGS BOLD
Here are examples of using .FIGURE and .FIGURE
DEFERRED. The results are clearer if you use the
/SEQUENCE qualifier when running off this file.
The page length for this example is 25 lines. The
width is 55 characters.
.BLANK
The following is Figure 1, a 3-line figure:
.BLANK
.CENTER;^*Title for Figure 1\*
.FIGURE 3
.SUBTITLE ^*Title for Figure 2\*
The next figure will be deferred to the following
page. It will be 7 lines long and will occur at
the top of page 2.
.FIGURE DEFERRED 7
.BLANK
This text occurs after Figure 1, still on page 1.
It demonstrates that text will continue to fill
the previous page after a .FIGURE DEFERRED
command is processed, but before it is triggered.
.PAGE
Now we are on page 2 after the occurrence of
Figure 2. We are about to set up Figure 3 to occur
on page 3. This time, we will use .PAGE followed
by
_.FIGURE 4. Page 2 will be left short.
.SUBTITLE ^*Title for Figure 3\*
.PAGE
.FIGURE 4
.SUBTITLE
And here is the text following Figure 3. It
appears physically after the figure on Page 3.
This is in contrast to the .FIGURE DEFERRED
behavior we saw for Figure 2; the text was not
pulled up to fill page 2.
```

The output file is this:

```
Here  are  examples  of  using  .FIGURE   and   .FIGURE
DEFERRED.   The  results  are  clearer  if  you use the
/SEQUENCE qualifier when running off  this  file.   The
page length for this example is 25 lines.  The width is
55 characters.

The following is Figure 1, a 3-line figure:

                   Title for Figure 1



The next figure will be deferred to the following page.
It  will  be  7 lines long and will occur at the top of
page 2.

This text occurs after Figure 1, still on page  1.   It
demonstrates  that  text  will  continue  to  fill  the
previous page  after  a  .FIGURE  DEFERRED  command  is
processed, but before it is triggered.

                           1
```

```
                   Title for Figure 2







Now we are on page 2, after the occurrence of Figure 2.
We  are  about  to  set up Figure 3 to occur on page 3.
This time, we will use .PAGE  followed  by  .FIGURE  4.
Page 2 will be left short.









                           2
```



Title for Figure 3

And here is the text following Figure 3. It appears physically after the figure on page 3. This is in contrast to the .FIGURE DEFERRED behavior we saw for Figure 2; the text was not pulled up to fill page 2.

3

.FILL and .NO FILL

Function:

The .FILL command causes DSR to treat line endings exactly like spaces (see also .NO SPACE). Line-filling is the accumulation of words on a line until the addition of one more word would exceed the right margin. If .NO FILL is in effect, line endings in the input file are duplicated in the output file (see also .KEEP).

Format:

Enable	Disable
.FILL	.NO FILL
.F	.NF

Characteristics:

- Both .FILL and .NO FILL issue .BREAK commands before doing their main tasks.
- The .NO FILL command suspends both line-filling and justification.
- The .FILL command restores line-filling and normally restores the most recent .JUSTIFY or .NO JUSTIFY setting that was in effect. A no-justify state that was set as a result of a .NO FILL command is not considered when DSR is determining the most recent setting. In other words, .NO FILL turns off both filling and justification, and .FILL restores them.
- .NO FILL suspends any .AUTOPARAGRAPH or .AUTOTABLE that has been issued; .FILL restores it.
- You can create an uneven right margin (ragged right text format) by having both .FILL and .NO JUSTIFY in effect.
- If you want justification of text without lines being filled, you must issue .NO FILL before you issue .JUSTIFY. Under these conditions, the same words appear on each line of the output file as were present in the input file. DSR inserts as much space between words as it needs to expand the line to the right margin.

Effects of other commands:

- The following commands issue .FILL commands:

  .APPENDIX
  .CHAPTER
  .HEADER LEVEL

- .NOTE also issues a .FILL, but .END NOTE restores the setting to what it was before you issued the .NOTE command.

Default:

.FILL

.FIRST TITLE

Function:

The .FIRST TITLE command allows running head information to appear on the first page of a document with no chapters. (See also .HEADERS ON, .LAYOUT, .TITLE, .SUBTITLE, and .AUTOSUBTITLE.)

Format:

```
.FIRST TITLE

.FT
```

Characteristics:

- Insert the .FIRST TITLE command before any text on the first page.

Effects of other commands:

- If you issue a .CHAPTER or .APPENDIX, .FIRST TITLE does not work; but if .LAYOUT is set to print page numbers (of any kind) at the bottom of pages (.LAYOUT 1 or 2 or 3), you get a page number on the first page, even if you did not issue .FIRST TITLE.

Default:

No title on the first page

.FOOTNOTE and .END FOOTNOTE

Function:

The .FOOTNOTE command places the text following it at the bottom of the current page if there is room. If there is not enough room for the entire footnote, DSR places it at the bottom of the next page.

The .END FOOTNOTE command ends the footnote and restores any case, fill, justify, spacing, or margin settings that you might have changed in the footnote text.

Format:

Enable	Disable
.FOOTNOTE [n]	.END FOOTNOTE
.FN [n]	.EFN

n

The number of lines the footnote will occupy. This argument is included only for compatibility with older versions of RUNOFF and is not necessary or recommended.

Characteristics:

- .FOOTNOTE does not provide a footnote symbol such as * or (1), or any separation from the main text just above the footnote.

  1. You can create a separator of 15 underscores (_) as follows:

     ```
     .FOOTNOTE
     .REPEAT 15"__"
     .BREAK
     ```

     Note that you must type two underscores because the first is functioning as the Accept flag (see Chapter 3).

  2. You can put an asterisk (*) before the text of the footnote by adding

     ```
     .LEFT MARGIN 2
     .INDENT -2;*#text
     .END FOOTNOTE
     ```

- DSR tries to put all footnotes at the bottom of the page on which they occur. If this is not possible -- if a footnote reference occurs too near the bottom of the page, or you issue more footnotes than there is room for -- DSR puts one or more of the footnotes at the bottom of the following page.

- DSR does not split a single footnote over two pages.

Effects of other commands:

- If you have issued a .NO PAGING command, all footnotes appear at the end of your document.

.HEADER LEVEL

Function:

The .HEADER LEVEL command allows you to specify both the header level and the header itself (section title). Successive .HEADER LEVEL commands, at various levels, number the headers sequentially. For example, if your current section is numbered 3.5.2.4 and your document has chapters, then sequential numbering occurs as follows: If, for your next .HEADER LEVEL, you issue a .HEADER LEVEL 3, that section is numbered 3.5.2.5. But if your next .HEADER LEVEL is a .HEADER LEVEL 2, then that section is numbered 3.5.3. (See also .NUMBER LEVEL, .SET LEVEL, .STYLE HEADERS, and .DISPLAY LEVELS.)

Header Levels for Documents without Chapters and with Chapters

	Nonchapter Sections	Chapter n Sections
.HEADER LEVEL 1	1.0	n.1
.HEADER LEVEL 2	1.1	n.1.1
.HEADER LEVEL 3	1.1.1	n.1.1.1

	Index Sections	Appendix Sections
.HEADER LEVEL 1	INDEX.1	A.1
.HEADER LEVEL 2	INDEX.1.1	A.1.1
.HEADER LEVEL 3	INDEX.1.1.1	A.1.1.1

Format:

```
               ┌+n┐
.HEADER LEVEL  │ n│ [title]
               └-n┘

    ┌+n┐
.HL │ n│ [title]
    └-n┘
```

n

A number from 1 to 6 that specifies the level of the header. Do not confuse the level numbers with the header numbers that are printed in your document just to the left of the header title.

+n

Adds n to the current header level number.

-n

Subtracts n from the current header level number.

title

The name of the section you are now starting. Do not precede the title with a semicolon (;) or include a semicolon in the title unless you type the Accept flag (_) just before it. A semicolon will terminate the command, causing the rest of the title to appear as ordinary text.

Characteristics:

- The .HEADER LEVEL command issues a .BREAK before doing its main task.
- .HEADER LEVEL issues a .TEST PAGE using seven more than the .PARAGRAPH or .SET PARAGRAPH value (you can change the test-page value by using the .STYLE HEADERS command). If the required number of lines is not available on the current page, DSR puts the header at the beginning of the next page. Note that these paragraph commands take into account the .SPACING value when they interpret the test-page value.
- .HEADER LEVEL issues a .FILL. It also issues a .JUSTIFY unless you have issued a .NO AUTOJUSTIFY. (However, note that if .JUSTIFY was in effect before you issued .HEADER LEVEL, .NO AUTOJUSTIFY does not cancel the .JUSTIFY.)
- .HEADER LEVEL 1 text is normally printed in all uppercase letters regardless of the case in which it is typed. All other levels are normally printed with only the first letter of each word in uppercase. You can override the initial capitalization of words like of and the by preceding any of them with a Lowercase flag (\). Alternatively, you can have the case exactly as you type it by preceding the title with the flag pair ^^. (See also .STYLE HEADERS.)
- Header levels 3 through 6 are normally run into the text, but separated from it by a hyphen (-). (See also .STYLE HEADERS.)
- If the title you specify for the header is wider than the current margins, it is filled (and justified, if justification is in effect) between the right margin and the header-level number. For example, the input shown in Example 2-6 produces the output shown in Example 2-7.

Example 2-6: Long Header Level (Input)

```
.RIGHT MARGIN 40
.HEADER LEVEL 2 Long header level that exceeds the margins
More text, which will revert to Column 1.
```

Example 2-7: Long Header Level (Output)

```
2.3  Long Header Level That Exceeds  The
     Margins

More text, which will revert  to  Column
1.
```

Effects of other commands:

- If .AUTOSUBTITLE is in effect, titles of specified header levels are used as running subheads at the tops of pages.
- You can use the .SET LEVEL command to preset the level of the next section head without issuing a previous .HEADER LEVEL.

Defaults:

If you issue .HEADER LEVEL without specifying a level number, you get the current header level. All header levels, .HEADER LEVEL 1 to .HEADER LEVEL 6, begin their numbering with 1 unless you specify another value with .NUMBER LEVEL.

.HEADERS ON

.HEADERS ON and .NO HEADERS

Function:

The .HEADERS ON and .NO HEADERS commands cancel and restore, respectively, the capability of having one or two lines of information at the top of a page. These lines indicate the content of the page and the page number. They are called running heads, which you should not confuse with section heads (specified with .HEADER LEVEL commands).

Format:

Enable	Disable
.HEADERS [ON]	.NO HEADERS
.HD [ON]	.NHD
.HD	

Characteristics:

- Standard running head information consists of a title at the top left of the page and the page number preceded by the word <u>page</u> at the top right (flush with the .PAGE SIZE page width, <u>not</u> with the right margin). A subtitle normally appears on the second line of the page immediately below the title. You can also have the current date appear on the right, immediately below the page number.

  Note that you can cause information to appear in positions other than these by issuing the .LAYOUT command.

- No running heads appear on the first page unless you issue a .FIRST TITLE command, but if you have issued a .LAYOUT command that puts page numbers at the bottom of pages, the first page will have a number even if you did not issue .FIRST TITLE.

- If you are using .CHAPTER commands in your document and headers are on, DSR uses the title of the current chapter as the running head title. You can, however, override this function by issuing a .TITLE command after the .CHAPTER command. If you are not using .CHAPTER commands, you must issue a .TITLE command to get a running head title.

- You use the .SUBTITLE command to specify a subtitle for running heads or to make it possible for you to issue a valid .AUTOSUBTITLE or .DATE. .AUTOSUBTITLE uses titles of .HEADER LEVEL commands (down to whatever level you specify) as subtitles in running heads.

- If you are using .CHAPTER, .APPENDIX, or .DO INDEX commands, page numbers appear in the form c-p, where c is the chapter number and p is the page number within the chapter. If your document has no chapters, you get page numbers of the form n. You can control the case of the word page by issuing a .HEADERS LOWER, .HEADERS UPPER, or .HEADERS MIXED command. You can suppress page numbers in the running heads by issuing .NO NUMBER.

- You can cause the current date to appear in the running heads by issuing the .DATE command. You must also have issued .SUBTITLE for .DATE to work.

Default:

```
.HEADERS ON
```

.HEADERS UPPER
.HEADERS LOWER
.HEADERS MIXED

.HEADERS UPPER, .HEADERS LOWER, and .HEADERS MIXED

Function:

The .HEADERS UPPER/LOWER/MIXED commands specify the case of the word page that precedes the page number. The commands produce, respectively, PAGE, page, and Page. In an index, these commands also affect the word index as part of the page number, for example, Page Index-3. The command normally takes effect on the next page.

Format:

```
.HEADERS UPPER       .HEADERS LOWER       .HEADERS MIXED

.HD UPPER            .HD LOWER            .HD MIXED
```

Default:

```
.HEADERS MIXED
```

.IF, .IFNOT, .ELSE, and .ENDIF

Function:

The .IF, .IF NOT, .ELSE, and .ENDIF commands (also known as the conditional commands) cause portions of a DSR file to be processed or not processed, according to conditions that you specify. The commands refer to the /VARIANT qualifier that you specify in the DSR command line. (See also /DEBUG=CONDITIONALS and .VARIABLE.)

The command line /VARIANT is of the form

/VARIANT=x or /VARIANT="$x_1,x_2,...x_n$"

where x is a name you give to a portion, or to separate portions, of a DSR file to be processed.

In the following definitions, A is a name that may have been specified in a /VARIANT qualifier.

Command	Meaning
.IF A	If A was specified in /VARIANT, then DSR processes the portion of the file following .IF A down to the next .ELSE A or .ENDIF A. If A was not specified in /VARIANT, then DSR ignores the portion of the file following .IF A down to the next .ELSE A or .ENDIF A.
.IFNOT A	If A was not specified in /VARIANT, then DSR processes the portion of the file following .IFNOT A down to the next .ELSE A or .ENDIF A. If A was specified in /VARIANT, then DSR ignores the portion of the file following .IFNOT A down to the next .ELSE A or .ENDIF A.
.ELSE A	If DSR processed the portion of the file following the most recent .IF A or .IFNOT A, it ignores the portion of the file following .ELSE A down to the next .ENDIF A. If DSR ignored the portion of the file following the most recent .IF A or .IFNOT A, it processes the portion of the file following .ELSE A down to the next .ENDIF A.
.ENDIF A	DSR terminates the most recent .IF A or .IFNOT A.

Format:

```
    .IF name

    .ELSE name

    .ENDIF name (or .EI name)

        or

    .IFNOT name (or .IN name)

    .ELSE name

    .ENDIF name
```

name

A word that is common to the conditional commands that make up a single .IF or .IFNOT block of text.

Characteristics:

- You can have .IF/.ELSE/.ENDIF blocks (or .IFNOT/.ELSE/.ENDIF blocks) within other such blocks, as shown in Example 2-8. (See similar discussion in description of .LIST and .END LIST.)
- You are not required to use an .ELSE command in an .IF or .IFNOT block.
- You cannot have an .IF and an .IFNOT with the same name in the same block.
- For purposes of debugging or analysis, you can distinguish those portions that would normally be processed from those that would not normally be processed. See .VARIABLE and /DEBUG=CONDITIONALS.
- If /VARIANT and /DEBUG are both present in the command line, /DEBUG overrides /VARIANT.

Example 2-8 shows how you enter three groups of conditional commands (A, B, C) with groups B and C nested within group A. This example explains what DSR does when it encounters each entry. Segment refers to a block of text.

Example 2-8: Conditional Commands

Entry	Explanation
.IF A	Begin segment A.
A1 Segment	Process A1 if A was specified.
.IF B	Begin segment B.
B1 Segment	Process B1 if both A and B were specified.
.ELSE B	Alternative segment B.
B2 Segment	Process B2 if A was specified and B was not.
.ENDIF B	End segment B.
A2 Segment	Process A2 if A was specified.
.ELSE A	The following segment is processed only if A was not specified.
A3 Segment	Process A3 if A was not specified.
.IF C	Begin segment C.
C1 Segment	Process C1 if A was not specified and C was.
.ELSE C	Alternative segment C.
C2 Segment	Process C2 if neither A nor C was specified.
.ENDIF C	End segment C.
A4 Segment	Process A4 if A was not specified.
.ENDIF A	End segment A.

Effects of other commands:

- If you issue a .VARIABLE command in your text file and /DEBUG=CONDITIONALS in the DSR command line (or just /DEBUG if you do not want to limit debugging to conditionals), all the segments in the example above will be processed. In addition, you can issue .VARIABLE commands in your text file to distinguish between the portions of text generated by the various segments.

.INDENT

.INDENT

Function:

The .INDENT command causes the first line of text following it to begin at a position relative to the left margin.

Format:

$.\mathrm{INDENT}\begin{bmatrix} n \\ -n \end{bmatrix}$

$.\mathrm{I}\begin{bmatrix} n \\ -n \end{bmatrix}$

n

specifies how many character positions to the right of the .LEFT MARGIN setting the first line of text will begin.

-n

specifies how many character positions to the left of the .LEFT MARGIN setting the first line of text will begin. An .INDENT cannot begin to the left of character position 0.

Characteristics:

- The .INDENT command issues a .BREAK before doing its main task.

Effects of other commands:

- If you issue a .BREAK immediately after a .INDENT, you will cancel the indentation just requested. This effect occurs with most of the commands that issue .BREAK commands.

Defaults:

If you issue .INDENT without a number, you get the indent value that you specified with .PARAGRAPH or .SET PARAGRAPH. If you did not issue either of these paragraph commands, you get an indentation of 5.

.INDEX

Function:

The .INDEX command creates an index entry with a page number reference. See Chapter 5, Creating an Index, for a more detailed discussion of this command.

Format:

.INDEX topic[>subtopic$_1$...>subtopic$_n$]

.X topic[>subtopic$_1$...>subtopic$_n$]

.JUSTIFY

.JUSTIFY and .NO JUSTIFY

Function:

The .JUSTIFY command causes DSR to insert exactly enough space between words to make lines reach the right margin (justification). The .NO JUSTIFY command disables justification.

Format:

Enable	Disable
.JUSTIFY	.NO JUSTIFY
.J	.NJ

Characteristics:

- Both .JUSTIFY and .NO JUSTIFY issue .BREAKs before doing their main tasks.
- You can create an uneven right margin (ragged right text format) by using the .NO JUSTIFY command and the .FILL command.
- If you want text justified without lines being filled, you must issue .JUSTIFY after you issue .NO FILL.

Effects of other commands:

- .NO FILL suspends justification as well as line-filling.
- .FILL normally restores the most recent .JUSTIFY or .NO JUSTIFY setting that was in effect. A .NO JUSTIFY state that was set as a result of .NO FILL is not considered when DSR is determining the most recent setting. In other words, .NO FILL turns off both filling and justification, and .FILL restores them.
- The following commands issue .JUSTIFY commands unless you have issued .NO AUTOJUSTIFY:

  .APPENDIX
  .CHAPTER
  .HEADER LEVEL
  .NOTE
- If you have issued .NO AUTOJUSTIFY and .JUSTIFY is in effect at the time you issue any of the above five commands, .JUSTIFY is not canceled (see .NO AUTOJUSTIFY).

Default:

.JUSTIFY

.KEEP and .NO KEEP

Function:

The .KEEP command allows you to keep in the output file blank lines that are present in the input file, when .NO FILL is in effect. Normally, multiple blank lines in the input file are discarded in the output file while .NO FILL is in effect. This is also the effect of .NO KEEP. (See also .LITERAL.)

Format:

Enable	Disable
.KEEP	.NO KEEP
.K	.NK

Characteristics:

- The .KEEP command differs from .LITERAL in that it allows you to issue commands and flags while it is in effect. For example, you can issue a .REQUIRE command after .KEEP and .NO FILL, whereas after .LITERAL no commands are recognized until after a .END LITERAL command.

Default:

.NO KEEP

Example:

Example 2-9 shows how to use the .KEEP command.

Example 2-9: .KEEP

The following shows the input text format, without the .KEEP command, before it is processed by DSR:

```
------------------------------------
-----------------------------------
-------------------------------------
.BLANK
--------------------------------------
-------------------------------------
.BLANK.NO FILL
  ------------------------------

  ----------------------------------


  -------------------------------
.BLANK.FILL
--------------------------------------
```

Here is the output text format. Note that the .NO FILL command does not cause DSR to keep the blank lines as they appear in the input text:

```
-------------------------------------------
-------------------------------------------
-------------------------------------------

-------------------------------------------
-------------------------------------------

  ------------------------------
  ----------------------------------
  -------------------------------

-------------------------------------------
```

Here is the same input file, with the .KEEP command added, before processing by DSR:

```
------------------------------------
-----------------------------------
-------------------------------------
.BLANK
--------------------------------------
-------------------------------------
.KEEP
.BLANK.NO FILL
  ------------------------------

  ----------------------------------


  -------------------------------
.BLANK.FILL
-------------------------------------------
```

The output file is this:

```
     -------------------------------------------
     -------------------------------------------
     -------------------------------------------

     -------------------------------------------
     -------------------------------------------

       ------------------------------

       ----------------------------------


       -------------------------------

     -------------------------------------------
```

Note that the .KEEP command causes DSR to keep the blank lines as they appear in the input file.

.LAYOUT

.LAYOUT

Function:

The .LAYOUT command rearranges running head information on pages. (See also .HEADERS ON.)

Format:

.LAYOUT n_1[,n_2]

.LO n_1[,n_2]

n_1

A number from 0 to 3 that specifies an alternative arrangement of running head information as follows:

.LAYOUT 0
: Restores the standard arrangement of a title and subtitle in the upper left of a page, and a page number and date in the upper right.

.LAYOUT 1
: Titles and subtitles are centered at the tops of pages. Page numbers are centered at the bottom.

.LAYOUT 2
: Titles and subtitles appear at the top right of right-hand (odd-numbered) pages and the top left of left-hand (even-numbered) pages. Page numbers are centered at the bottom.

.LAYOUT 3
: Gives the standard page arrangement for title and subtitle (as in .LAYOUT 0), but with the addition of running page numbers centered at the bottom of pages between two hyphens (for example, - 23 -). Running page numbers are consecutive through the entire document rather than within chapters; they are not affected by the .NO NUMBER or .NUMBER PAGE command. (But see .NUMBER RUNNING.)

n_2

Specifies how many lines below the last line of text on a page the number will appear. You must specify n_2 if n_1 is 1, 2, or 3. If n_1 is 0, you cannot specify a value for n_2.

Characteristics:

- The .LAYOUT command issues a .BREAK before doing its main task.
- The page length (.PAGE SIZE) is not affected by this command.

Default:

.LAYOUT 0

Examples:

Examples 2-10 through 2-14 below are for pages of a second chapter for a document with 20 pages in the first chapter. The .DATE command has been issued. The title is "INTRODUCTION TO DSR"; the subtitle is "DSR COMMAND FORMAT."

Example 2-10: .LAYOUT 0 (Default)

```
INTRODUCTION TO DSR                                         Page 2-2
DSR COMMAND FORMAT                                         14 Dec 81

                                .
                                .
                                .
                       [text of page 2-2]
                                .
                                .
                                .
```

Example 2-11: .LAYOUT 1,2

```
                       INTRODUCTION TO DSR
                        DSR COMMAND FORMAT

                                .
                                .
                                .
                       [text of page 2-2]
                                .
                                .
                                .

                               2-2
```

Example 2-12: .LAYOUT 2,2 (Even Page)

```
INTRODUCTION TO DSR
DSR COMMAND FORMAT
                                .
                                .
                                .
                       [text of page 2-2]
                                .
                                .
                                .

                               2-2
```

Example 2-13: .LAYOUT 2,2 (Odd Page)

```
                                                  INTRODUCTION TO DSR
                                                   DSR COMMAND FORMAT

                                  .
                                  .
                                  .
                         [text of page 2-3]
                                  .
                                  .
                                  .

                                 2-3
```

Example 2-14: .LAYOUT 3,2

```
INTRODUCTION TO DSR                                          Page 2-2
DSR COMMAND FORMAT                                          14 Dec 81

                                  .
                                  .
                                  .
                         [text of page 2-2]
                                  .
                                  .
                                  .

                                -22-
```

.LEFT MARGIN

Function:

The .LEFT MARGIN command sets the left margin to the specified position.

Format:

$$.\text{LEFT MARGIN}\begin{bmatrix} +n \\ n \\ -n \end{bmatrix}$$

$$.\text{LM}\begin{bmatrix} +n \\ n \\ -n \end{bmatrix}$$

n

Specifies the number of the character position of the new left margin. For example, .LEFT MARGIN 0 sets the left margin just to the left of the first character position.

+n

Sets the left margin n character positions to the right of the current left margin.

-n

Sets the left margin n character positions to the left of the current left margin.

Characteristics:

- .LEFT MARGIN issues a .BREAK before setting the left margin.
- The value for n must be smaller than the right margin value.
- The resulting value for +n must be smaller than the right margin value (see .RIGHT MARGIN).
- The resulting value for -n cannot be less than 0.

Effects of other commands:

- .LIST and .NOTE change the left margin; .END LIST and .END NOTE restore it.

Default:

.LEFT MARGIN 0

.LIST

.LIST and .END LIST

Function:

The .LIST command specifies the beginning of a list by resetting the .LEFT MARGIN farther to the right, setting a .SKIP value to take effect before each item in the list, and issuing a .TEST PAGE. You use a .LIST ELEMENT command to specify each item in the list. .LIST ELEMENT commands also give you numbers or letters in sequence in the left margin, or let you substitute a single character of your choice for each of the numbers or letters (for example, the lowercase letter o, which is known as a "bullet"). (See also .DISPLAY ELEMENTS and .NUMBER LIST.)

The .END LIST command ends a list, restoring any fill, justify, case, margin, or spacing settings that were in effect before you issued the most recent .LIST command. You can also specify a value with .END LIST that puts blank lines after the last item in the list (as with .SKIP).

Format:

Begin	**End**
.LIST $\begin{bmatrix} n \\ -n \end{bmatrix}$ ["x"]	.END LIST $\begin{bmatrix} n \\ -n \end{bmatrix}$
or	or
.LIST $\begin{bmatrix} n \\ -n \end{bmatrix}$ ['x']	.ELS $\begin{bmatrix} n \\ -n \end{bmatrix}$
.LS $\begin{bmatrix} n \\ -n \end{bmatrix}$ $\begin{bmatrix} "x" \\ 'x' \end{bmatrix}$	

n

With .LIST, specifies the number of blank lines to appear before each item, but each blank line can result in additional blank lines if the .SPACING setting is greater than 1. (See .SKIP n.)

-n

With .LIST, pushes the next line of list text to within n lines of the bottom of the current page by inserting blank lines. (See .SKIP -n and .BLANK -n.)

n

With .END LIST, behaves the same as n with .LIST. However, the blank lines appear after the final item in the current list.

-n

With .END LIST, pushes the next line of current list text to within n lines of the bottom of the current page by inserting blank lines. (See .SKIP -n and .BLANK -n.)

x

A character enclosed by quotation marks ("x") or apostrophes ('x') that you can specify to appear at the beginning of each item.

Characteristics:

- A list is defined by a .LIST and a paired .END LIST. Other lists, similarly defined, can exist within it.

- If the .LEFT MARGIN is set at 0 before you issue .LIST, the .LIST issues a .LEFT MARGIN 9. If the .LEFT MARGIN is not set at 0 just before you issue .LIST, then the .LIST issues a .LEFT MARGIN +4.

- .LIST issues a .TEST PAGE, adding 2 to the test-page value from either the most recently issued .PARAGRAPH or .SET PARAGRAPH command or from its default value. Paragraph commands take into account the .SPACING value that is in effect when they interpret the test-page value.

- You can issue a .LIST without having to issue a .LIST ELEMENT for it.

- You must pair every .LIST you issue with an .END LIST. It is important to know how the pairing works if you want to make lists within other lists:

  1. While you are in a list you can issue a new .LIST without having issued an .END LIST for the previous .LIST.

  2. When you issue the next .END LIST, you end only the new inner .LIST.

  3. You can then add more items to the main outer .LIST by issuing more .LIST ELEMENT commands. All attributes of the outer list will be restored, including the resumption of any sequence of numbers or letters that was in effect. For example:

```
.LIST (main)
.LIST ELEMENT
.LIST ELEMENT

.LIST (new)
.LIST ELEMENT
.LIST ELEMENT

.END LIST (new)

.LIST ELEMENT
.LIST ELEMENT

.END LIST (main)
```

  You can also issue another .END LIST to end the main list immediately after the .END LIST that ends the new inner list.

Defaults:

- If you omit n from either .LIST or .END LIST, you get the .SKIP value associated with .PARAGRAPH or .SET PARAGRAPH or 1 if you have not specified such a value.

- If you omit "x," you get a sequence of decimal numbers, beginning with 1, or you get another kind of sequence if you issue a .DISPLAY ELEMENTS command after the .LIST.

.LIST ELEMENT

.LIST ELEMENT

Function:

The .LIST ELEMENT command specifies the beginning of each item in a list. If you specify a character in a .LIST command, it appears, followed by two spaces, before each item. Otherwise, a sequence of numbers or letters as defined in the .DISPLAY ELEMENTS command appears when you issue successive .LIST ELEMENT commands. If you have not issued a .DISPLAY ELEMENTS command, you will get a sequence of numbers, each followed by a period and two spaces. (See .LIST, .END LIST, .DISPLAY ELEMENTS, and .NUMBER LIST.)

Format:

.LIST ELEMENT

.LE

Characteristics:

- The .LIST ELEMENT command issues a .BREAK before doing its main task.
- You can specify the number of the next element in the sequence by issuing a .NUMBER LIST.
- A sequence is interrupted by the next occurrence of a .LIST or .END LIST.
- If you have not yet ended the current list, a new .LIST temporarily suspends all the attributes of the old one. The next .END LIST ends only the new .LIST and restores all the attributes of the old .LIST.
- If you have issued a .LIST command, both the number of blank lines before each .LIST ELEMENT item and the .LEFT MARGIN setting for the list of items are as noted in the .LIST description.
- You can issue .LIST ELEMENT without having issued .LIST. If you have not issued a .LIST command, you will get the .LEFT MARGIN setting that was in effect before you issued the .LIST ELEMENT. If an item is more than one line long, the character or number normally printed in the left margin will be merged with the text of the item instead.

Defaults:

- If you have not issued a .LIST command with a specified character to appear in the left margin, you will get the kind of sequence that you specified in .DISPLAY ELEMENTS.
- If you have not issued a .DISPLAY ELEMENTS command, you will get decimal numbers, each followed by a period and two spaces.

.LITERAL

Function:

The .LITERAL command allows you to have your text formatted exactly as you have typed it. This means that you will get a blank line in the output file wherever a blank line occurs in the input file. Commands are not recognized when .LITERAL is in effect and are treated as ordinary text if you enter them. DSR flags are also treated as normal text. Tab stops set prior to the .LITERAL command, however, are still in effect within the block of .LITERAL text (see .TAB STOPS). You must issue an .END LITERAL when you want DSR to resume normal formatting.

Format:

Enable	Disable
.LITERAL [n]	.END LITERAL
.LT [n]	.EL

n

The number of lines to be produced. This argument is included only for compatibility with older versions of RUNOFF and is not necessary or recommended.

Characteristics:

- .LITERAL issues a .BREAK before doing its main task.
- .LITERAL issues a .RIGHT MARGIN 150.
- If .LITERAL is in effect, a blank line occurs in the output file wherever one occurs in the input file. If .LITERAL (or .KEEP) is not in effect, you cannot produce blank lines by putting them in the input file, even if .NO FILL is in effect.
- Between .LITERAL and .END LITERAL, all recognition of commands and flags is suspended. In addition, nearly all commands and flags that were in effect before you issued .LITERAL are disabled. For example, previously specified case, fill, and justify operations are disabled. If you want to be able to use commands and flags and still get all the benefits of .LITERAL, you can use the .KEEP and .NO FILL commands instead of .LITERAL.
- The settings that were in effect for the following commands and flags prior to the .LITERAL remain in effect for the .LITERAL text:

  .LEFT MARGIN

  .TAB STOPS

  ^& (Underlining)

  ^* (Boldfacing)

.NO SPACE

.NO SPACE

Function:

The .NO SPACE command prevents the insertion of the space for one text line only, causing the characters at the end of one line and the beginning of the next to be adjacent.

Without the .NO SPACE command, when .FILL is in effect, DSR treats the end of an input line exactly like a space. That is, it inserts a space in the output file at the place where each input line ended (this is the meaning of "fill").

If you ever have occasion to use this command, you should issue it immediately after the end-of-line that you want to affect.

Format:

```
.NO SPACE

.NSP
```

Default:

You get the normal space for the RETURN unless you issue .NO SPACE.

Example:

Example 2-12 illustrates the use of the .NO SPACE command. The example contains two files. The first file shows what occurs when you do not use the .NO SPACE command and the second shows what occurs when you do use the .NO SPACE command.

Example 2-15: .NO SPACE

The input of the first file, in which there are no .NO SPACE commands, is this:

```
     The following diagram illustrates the 16
     -bit, bit-slice, byte-encoded configuration:
     configuration:
```

The output file is this:

```
     The following diagram illustrates the  16  -bit,  bit-slice,
     byte-encoded configuration:
```

Notice the space between "16" and "-bit." To avoid having this space, use the .NO SPACE command. The next input file shows this:

```
     The following diagram illustrates the 16
     .NO SPACE
     -bit, bit-slice, byte-encoded configuration:
```

The processed output file looks like this:

```
     The following diagram  illustrates  the  16-bit,  bit-slice,
     byte-encoded configuration:
```

Notice that now there is no space between "16" and "-bit."

.NOTE

.NOTE and .END NOTE

Function:

The .NOTE command highlights a portion of text by narrowing the margins and printing a title centered over the text.

The .END NOTE command restores the fill, justify, case, margin, and spacing settings that were in effect just before you issued the .NOTE.

Format:

Begin	End
.NOTE [text]	.END NOTE $\begin{bmatrix} n \\ -n \end{bmatrix}$
.NT [text]	.EN $\begin{bmatrix} n \\ -n \end{bmatrix}$

n

Specifies the number of blank lines to follow the note. If .SPACING has a value greater than 1, you will get more lines than you specified. (See also .SKIP n.)

-n

Specifies that the next line of text be pushed to within n lines of the bottom of the current page by the insertion of blank lines. (See also .SKIP -n and .BLANK -n.)

text

A title for the note. If omitted, you get the word NOTE.

Characteristics:

- The .NOTE command issues a .BREAK before doing its main task.
- .NOTE does a .TEST PAGE, taking the test-page value from the most recently issued .PARAGRAPH or .SET PARAGRAPH command and adding 4 to it. Paragraph commands take into account the .SPACING value when they interpret the test-page value. (See also .SKIP.)
- .NOTE then issues a .SKIP 2 above the title and a .SKIP 1 below the title.
- If the left margin is set to 0, .NOTE issues a .LEFT MARGIN +15 and a .RIGHT MARGIN -15. Otherwise, the margins are set +4 and -4, respectively. (See .LEFT MARGIN and .RIGHT MARGIN.)
- .NOTE issues a .FILL. It also issues a .JUSTIFY unless you have issued a .NO AUTOJUSTIFY. (However, note that if .JUSTIFY was in effect before you issued the .NOTE, .NO AUTOJUSTIFY does not cancel the .JUSTIFY.)
- .END NOTE issues a .SKIP 2 after the text and then restores all settings that were in effect before the .NOTE.

Default:

The word NOTE appears over the text if you do not specify a title. .END NOTE leaves 1 blank line after the note.

.NUMBER APPENDIX

.NUMBER APPENDIX

Function:

The .NUMBER APPENDIX command allows you to specify an identifying letter with which a sequence of appendixes will begin. The next .APPENDIX command starts the sequence. Subsequent .APPENDIX commands cause appendixes to be lettered in alphabetic order.

Format:

.NUMBER APPENDIX $\begin{Bmatrix} +n \\ 1 \\ -n \end{Bmatrix}$

.NMA $\begin{Bmatrix} +n \\ 1 \\ -n \end{Bmatrix}$

1

The letter that the next .APPENDIX will have. You can also specify a number corresponding (in order) to the letter that the appendix will have. For example, 1=A, 26=Z, 27=AA, 28=AB.

+n

Specifies how many alphabetically ordered letters past the current appendix letter the next .APPENDIX letter will be. For example, if the current appendix is APPENDIX B, then .NUMBER APPENDIX +2 will cause the next .APPENDIX to produce APPENDIX D.

-n

Specifies how many alphabetically ordered letters before the current appendix letter the next .APPENDIX letter will be.

Characteristics:

- You can specify a string of up to 5 letters instead of a single letter. An appendix of examples, for instance, could be numbered EXAM and appear in a running head as Page EXAM-1,EXAM-2,...

Default:

Sequential uppercase lettering, beginning with A

.NUMBER CHAPTER

Function:

The .NUMBER CHAPTER command allows you to specify the chapter number with which a sequence of chapters will begin. The next .CHAPTER command starts the sequence. Subsequent .CHAPTER commands will cause each chapter to be numbered one higher than the chapter that precedes it. (See also .DISPLAY CHAPTER.)

Format:

$$\text{.NUMBER CHAPTER} \left\{ \begin{matrix} +n \\ n \\ -n \end{matrix} \right\}$$

$$\text{.NMCH} \left\{ \begin{matrix} +n \\ n \\ -n \end{matrix} \right\}$$

n

The number that the next .CHAPTER will have.

+n

Adds n to the number of the most recently issued .CHAPTER. The result is the number that the next .CHAPTER will have.

-n

Subtracts n from the number of the most recently issued .CHAPTER. The result is the number that the next .CHAPTER will have.

Default:

Sequential decimal numbering, beginning with 1

.NUMBER LEVEL

.NUMBER LEVEL

Function:

The .NUMBER LEVEL command allows you to specify the beginning number of a sequence of headers. You issue this command immediately before the first .HEADER LEVEL that you want to affect. Subsequent .HEADER LEVEL commands will each be one higher than the preceding one according to its level (see .HEADER LEVEL). (See also .STYLE HEADERS and .DISPLAY LEVELS.)

Header Levels for Documents without Chapters and with Chapters

	Nonchapter Sections	Chapter n Sections
.HEADER LEVEL 1	1.0	n.1
.HEADER LEVEL 2	1.1	n.1.1
.HEADER LEVEL 3	1.1.1	n.1.1.1

Format:

$$\text{.NUMBER LEVEL} \begin{bmatrix} +n_1 \\ n_1 \\ -n_1 \end{bmatrix}, \begin{bmatrix} +n_2 \\ n_2 \\ -n_2 \end{bmatrix}, \ldots \begin{bmatrix} +n_6 \\ n_6 \\ -n_6 \end{bmatrix}$$

$$\text{.NMLV} \begin{bmatrix} +n_1 \\ n_1 \\ -n_1 \end{bmatrix}, \begin{bmatrix} +n_2 \\ n_2 \\ -n_2 \end{bmatrix}, \ldots \begin{bmatrix} +n_6 \\ n_6 \\ -n_6 \end{bmatrix}$$

$n_1, n_2, \ldots n_6$

Indicate positioned numbers in a section header. The commas correspond to the dots in the printed section number. For example, to set the next .HEADER LEVEL to 3.5.2.4, you would issue

```
.NUMBER LEVEL 3,5,2,4

.HEADER LEVEL
```

$+n_1$

Adds n to the current value of n_1.

$-n_1$

Subtracts n from the current value of n_1.

Characteristics:

- When you issue the first .HEADER LEVEL after a .NUMBER LEVEL, do not specify a level number for it (note example above). If you specify a level number that does not correspond to the level you indicated in the .NUMBER LEVEL command, you will get unexpected results.

Default:

Sequential decimal numbering, beginning with 1

.NUMBER LIST

Function:

The .NUMBER LIST command allows you to specify, anywhere in a list, the number with which a sequence of items in a list will begin. You issue this command just before the next .LIST ELEMENT that you want to affect. Subsequent list elements will each have a number that is one greater than the number for the preceding .LIST ELEMENT. (See also .DISPLAY ELEMENTS, with which you can specify the form the number will take.)

Format:

.NUMBER LIST [n_1,]n_2

.NMLS [n_1,]n_2

n_1

A reference to particular lists. This argument is included only for compatibility with older versions of RUNOFF and is not necessary or recommended.

n_2

Specifies a string of characters or the first number of a sequence identifying items in a list. You must issue a .LIST ELEMENT for each item.

Characteristics:

- A list is defined by a .LIST and a paired .END LIST. Other lists, similarly defined, can exist within it.

Default:

Sequential decimal numbering, beginning with 1

.NUMBER PAGE

.NUMBER PAGE and .NO NUMBER

Function:

The .NO NUMBER command suspends normal page numbering. The .NUMBER PAGE command resumes normal page numbering, having kept track of the numbering while .NO NUMBER was in effect; or it allows you to specify the beginning of a new number sequence by specifying a number for the next page. (See also .NUMBER RUNNING, .DISPLAY NUMBER, .NO PAGING, and .HEADERS ON.)

Format:

```
         Enable                      Disable

                      ┌+n┐
.NUMBER [PAGE]│ n │          .NO NUMBER
                      └-n┘

        ┌+n┐
.NMPG│ n │                   .NNM
        └-n┘
```

n

The number that the next page will have.

+n

Sets the number of the next page to n more than the number of the current page.

-n

Sets the number of the next page to n less than the number of the current page.

Characteristics:

- If you are using .CHAPTER, .APPENDIX, or .DO INDEX commands, page numbers appear in the form c-p, where c is the chapter number and p is the page number within the chapter. If your document has no chapters, you get page numbers of the form n.
- .NO NUMBER and .NUMBER PAGE do not affect running page numbers (that is, the numbering you get by issuing .LAYOUT with an nl value of 3). (See also .NUMBER RUNNING.)

Default:

Sequential decimal numbering, beginning with 1 or chapter-1

.NUMBER RUNNING

Function:

The .NUMBER RUNNING command allows you to specify the beginning of a new sequence of running page numbers. This command affects only the page numbers at the bottom of a page that you specified by issuing a .LAYOUT command with an n^1 value of 3. (See .LAYOUT, .HEADERS ON, and .NO NUMBER.)

Format:

$$.\text{NUMBER RUNNING}\begin{Bmatrix} +n \\ n \\ -n \end{Bmatrix}$$

$$.\text{NMR}\begin{Bmatrix} +n \\ n \\ -n \end{Bmatrix}$$

n

The running number that the next page will have.

+n

Sets the running number of the next page to n more than the running number of the current page.

-n

Sets the running number of the next page to n less than the running number of the current page.

Characteristics:

- Running pages are produced only by .LAYOUT 3,n. They appear at the bottom of pages, are enclosed by hyphens (- n -), and run consecutively over the entire document. (Note that you can get similar numbering [without hyphens and including page 1] if you issue .LAYOUT 1,n or 2,n -- provided you are not using .CHAPTER commands. This numbering is not affected by the .NUMBER RUNNING command.)

Default:

No running page numbers

.NUMBER SUBPAGE

.NUMBER SUBPAGE

Function:

The .NUMBER SUBPAGE command allows you to specify the beginning of a new sequence of subpage numbers, for example, 1-16A, 1-16B, 1-16C, and so on. This command affects only the letter(s) that the .SUBPAGE command appends to the normally numeric page number. .NUMBER SUBPAGE takes effect on the next page. (See also .SUBPAGE and .DISPLAY SUBPAGE.)

Format:

$$.\text{NUMBER SUBPAGE}\begin{bmatrix}+n\\n\\-n\end{bmatrix}$$

$$.\text{NMSPG}\begin{bmatrix}+n\\n\\-n\end{bmatrix}$$

n

The subpage letter that will be appended to the number of the next page. You can also specify a number corresponding to the letter that the subpage will have. For example, 1=A, 26=Z, 27=AA, 28=AB.

+n

Specifies how many alphabetically ordered letters past the current subpage letter the next subpage letter will be. For example, if the current subpage is page 3-12E, then .NUMBER SUBPAGE +2 will cause the next subpage to be numbered 3-12G.

-n

Specifies how many alphabetically ordered letters before the current subpage letter the next subpage letter will be.

Characteristics:

- .NUMBER SUBPAGE turns on .SUBPAGE if it was not previously in effect.

Default:

Sequential uppercase lettering, beginning with A

.PAGE

Function:

The .PAGE command starts a new page.

Format:

```
.PAGE

.PG
```

Characteristics:

- The .PAGE command issues a .BREAK before doing its main task.
- The current page must have at least one line of text for .PAGE to work, so you cannot use .PAGE to generate a series of blank pages.
- You can generate a series of blank pages by alternately issuing .PAGE and .FIGURE commands.
- If .NO PAGING is in effect, you can still issue a .PAGE successfully, but you will not get additional pages as you normally would if the text on a page exceeds the page length associated with .PAGE SIZE.

.PAGE SIZE

.PAGE SIZE

Function:

The .PAGE SIZE command specifies the maximum number of lines of text on a page (page length) and the maximum number of characters on a line (page width). (Compare with .RIGHT MARGIN.) The width component of .PAGE SIZE and the .RIGHT MARGIN value are separate values. The .RIGHT MARGIN value never exceeds the .PAGE SIZE value.

Format:

$$\text{.PAGE SIZE} \quad \begin{bmatrix} +n_1 \\ n_1 \\ -n_1 \end{bmatrix} \begin{bmatrix} ,+n_2 \\ ,\ n_2 \\ ,-n_2 \end{bmatrix}$$

$$\text{.PS} \quad \begin{bmatrix} +n_1 \\ n_1 \\ -n_1 \end{bmatrix} \begin{bmatrix} ,+n_2 \\ ,\ n_2 \\ ,-n_2 \end{bmatrix}$$

n_1

(Length) is the maximum number of lines on a page. n_1 cannot be smaller than 13.

$+n_1$

Increases the current page length by n_1 lines.

$-n_1$

Decreases the current page length by n_1 lines.

n_2

(Width) is the maximum number of characters on a line. n_2 cannot be larger than 150.

$+n_2$

Increases the current page width by n_2 characters.

$-n_2$

Decreases the current page width by n_2 characters.

Characteristics:

- The .PAGE SIZE command issues a .BREAK before doing its main task.
- Running heads line up on the right with the page width setting.
- .PAGE SIZE sets the right margin to the page width, regardless of its previous value.
- If you have issued a .NO PAGING command, .PAGE SIZE restores the .PAGING state.

Effects of other commands:

- If you issue .RIGHT MARGIN with a value that is larger than the .PAGE SIZE width value, the .PAGE SIZE width value automatically increases to match the .RIGHT MARGIN value.

Default:

```
.PAGE SIZE 58,60
```

.PAGING

.PAGING and .NO PAGING

Function:

The .PAGING command enables paging. The .NO PAGING command disables it.

Format:

Enable	Disable
.PAGING	.NO PAGING
.PA	.NPA

Characteristics:

- .PAGING issues a .BREAK before doing its main task.
- If you have issued .NO PAGING, the document is not split into numbered pages and space is not reserved for running heads. Any mechanical page breaks that a print device causes still occur.
- If .NO PAGING is in effect and you want to start a new page after some lines of text, you can use the .PAGING command.
- If your input file has a file type of .RNH (Help file), paging is turned off and the .PAGING command has no effect.

Effects of other commands:

- If you issue .CHAPTER, .APPENDIX, or .PAGE SIZE you get .PAGING.
- You can temporarily override a .NO PAGING state by issuing a .PAGE, but unless you issue .PAGING, you will not get additional pages as you normally would if the text on a page exceeded the page length value of .PAGE SIZE.

Default:

.PAGING

.PARAGRAPH

Function:

The .PARAGRAPH command controls spacing and page placement associated with the creation of paragraphs. It issues a .TEST PAGE, followed by a .SKIP and an .INDENT. (See also .SET PARAGRAPH.)

Format:

$$.\text{PARAGRAPH} \left[\begin{bmatrix} n_1 \\ -n_1 \end{bmatrix}, \begin{bmatrix} n_2 \\ -n_2 \end{bmatrix} [,n_3] \right]$$

$$.\text{P} \left[\begin{bmatrix} n_1 \\ -n_1 \end{bmatrix}, \begin{bmatrix} n_2 \\ -n_2 \end{bmatrix} [,n_3] \right]$$

n_1

(.INDENT) specifies how many character positions to the right of the .LEFT MARGIN setting the first line of text will begin.

$-n_1$

Specifies how many character positions to the left of the .LEFT MARGIN setting the first line of text will begin. $-n_1$ cannot, however, cause the text to begin to the left of character position 0.

n_2

(.SKIP) is the number of blank lines you want inserted before the paragraph. You get additional blank lines if the .SPACING value is greater than 1 (see .SKIP).

$-n_2$

Specifies that the next line of text be pushed to within n_2 lines of the bottom of the current page by the insertion of blank lines. Every line but the last one retains the line spacing (.SPACING value) that follows it.

n_3

(.TEST PAGE) is the number of lines of text required to be on one page. n_3, unlike the .TEST PAGE command itself, takes into account any blank lines that .SPACING is routinely inserting after each line of text. If there is not enough room on the current page to accommodate that many lines, DSR puts the text on the next page. You can cancel this function by specifying 0 for n3.

Characteristics:

- The .PARAGRAPH command issues a .BREAK before doing its main task.

Defaults:

- If you issue .PARAGRAPH without one or more of the n values, you get the corresponding setting from the previous .PARAGRAPH or .SET PARAGRAPH that you issued.
- If you have not set values in any previous .PARAGRAPH or .SET PARAGRAPH that you might have issued, you get one or more of the following:

  $n_1=5$

  $n_2=1$

  $n_3=2$

Examples of default formats:

Format	Actual arguments
.PARAGRAPH ,,4	5,1,4
.PARAGRAPH 3	3,1,4
.PARAGRAPH ,2	3,2,4
.PARAGRAPH	3,2,4

.PERIOD and .NO PERIOD

Function:

DSR normally adds an extra space after any of the following punctuation marks in your text: period (.), semicolon (;), colon (:), question mark (?), and exclamation point (!).

The .NO PERIOD command cancels the extra space that DSR inserts after any of the punctuation marks listed above.

The .PERIOD command restores the routine insertion of an extra space following any of the punctuation marks listed above.

Format:

Enable	Disable
.PERIOD	.NO PERIOD
.PR	.NPR

Characteristics:

- The .PERIOD condition works as described only if .FILL is in effect and if the punctuation mark is followed by either a space or the end of the line.
- You can override the effects of a .PERIOD at any time by using the ACCEPT flag (_) in front of a punctuation mark.

Effects of other commands:

- The Period flag (the plus sign [+] by default) allows you to specify extra space after any other character than the ones mentioned above (see the description of the Period flag in Chapter 3).

Default:

.PERIOD

Example:

The .NO PERIOD command is used to differentiate between punctuation used in the syntax of a sentence and punctuation used for other purposes.

Example 2-16 illustrates the use of the .PERIOD command.

Example 2-16: .PERIOD

The following shows the input file before it is processed by DSR:

```
.FILL.NO JUSTIFY
.NO PERIOD
Mr. Jones, the ratios of 5! (factorial) to 6! is 5 : 6 but
we will not use the ? and ; characters.
.PERIOD
Wait a minute! The answers are: yes and no;  however, what
if you are not sure? Try not to confuse things.  Be more
explicit.
```

The output file is this:

```
Mr. Jones, the ratio of 5! (factorial) to 6! is 5 : 6 but we
will not use the ? and ; characters. Wait a minute!  The
answers are:  yes and no;  however, what if you are not
sure?  Try not to confuse things.  Be more explicit.
```

Notice that in the first sentence there is only one space after each punctuation mark. In the second sentence, where punctuation marks are used as usual, there are two spaces after each one (except the commas, which have one space after them).

The following shows the output of the same file with the .PERIOD command used throughout:

```
Mr.  Jones, the ratio of 5!  (factorial) to 6!  is 5 :  6
but we will not use the ?  and ;  characters.  Wait a
minute!  The answers are:  yes and no;  however, what if you
are not sure?  Try not to confuse things.  Be more explicit.
```

Notice that there are two spaces after each punctuation mark (except the commas).

The following shows the output of the same file with the .NO PERIOD command used throughout:

```
Mr. Jones, the ratio of 5! (factorial) to 6! is 5 : 6 but we
will not use the ? and ; characters. Wait a minute! The
answers are: yes and no; however, what if you are not sure?
Try not to confuse things. Be more explicit.
```

Notice that there is one space after each punctuation mark.

.PRINT INDEX

Function:

The .PRINT INDEX command is the same as the .DO INDEX command, except that it does not issue a new page. See Chapter 5, Creating an Index, for a more detailed discussion.

Format:

```
.PRINT INDEX

.PX
```

.REPEAT

Function:

The .REPEAT command allows you to specify up to 50 characters to be printed a specified number of times, either horizontally or vertically.

Format:

```
.REPEAT n "x" (or n 'x')

.RPT n "x" (or n 'x')
```

n

The number of times you want the characters printed.

x

A string of up to 150 characters. You must enclose the characters within either quotation marks (" ") or apostrophes (' ').

Characteristics:

- If you issue .REPEAT with .FILL in effect, the characters are repeated horizontally. If you want the repetitions separated by a space, you must include a space at the end of the characters to be repeated.
- If you issue .REPEAT with .NO FILL in effect, the characters are repeated vertically beginning at the left margin.

.REQUIRE

Function:

The .REQUIRE command allows you to process several DSR files at the same time and merge them in the output file.

Format:

```
.REQUIRE "file-spec" (or 'file-spec')

.REQ "file-spec" (or 'file-spec')
```

file-spec

A file specification enclosed in quotation marks or apostrophes. If you do not supply a full file specification, DSR fills in missing parts from your default device and directory. The default file type is .RNO.

Characteristics:

- A .REQUIRE command must be the last command on the line.
- You can have several .REQUIRE commands in a single file or you can have one or more .REQUIRE commands in files that are themselves specified in .REQUIRE commands. The maximum depth for "nesting" .REQUIRE files in this manner is 10.
- If you plan to use .REQUIRE extensively with several files in various combinations, see the description of the .SET LEVEL command, with which section heads in these files can automatically be adjusted to the right level in any context. (See also .HEADER LEVEL.)

.RIGHT

.RIGHT

Function:

The .RIGHT command positions a single line of text relative to the right margin. (See also .CENTER.)

Format:

$\texttt{.RIGHT}\begin{bmatrix} \texttt{n} \\ \texttt{-n} \end{bmatrix}\texttt{;text}$ $\qquad$ $\texttt{.RIGHT}\begin{bmatrix} \texttt{n} \\ \texttt{-n} \end{bmatrix}$
text

$\texttt{.R}\begin{bmatrix} \texttt{n} \\ \texttt{-n} \end{bmatrix}\texttt{;text}$ $\qquad$ $\texttt{.R}\begin{bmatrix} \texttt{n} \\ \texttt{-n} \end{bmatrix}$
text

n

Specifies how many character positions to the left of the right margin setting the line will be indented.

-n

Specifies the number of character positions to the right of the right margin setting that the line will extend to.

text

The text to be positioned relative to the right margin. No other DSR commands can follow this text on a line.

Characteristics:

- The .RIGHT command issues a .BREAK before doing its main task.
- The line of text being positioned can extend past margin settings and even beyond the .PAGE SIZE width setting, but it cannot go to the left of character position 0.
- You can enter the text to be positioned on the same line following the .RIGHT command. Or you can end the line after the .RIGHT command, in which case the following line of text is positioned.
- No commands will be recognized on the line following a .RIGHT command (or, if it is blank, on the next line). The control flag (.) is not honored while .RIGHT is collecting text to position. Other DSR flags are recognized, however, for example, the bold and the underline flags.
- If the text being positioned contains a semicolon (;), you must precede it with the Accept flag (_). Otherwise, DSR stops collecting text to position at the semicolon, issues an error message, and discards the rest of the line.

Default:

If you issue .RIGHT without a value, you get a 0, which will push the line of text to the right margin.

.RIGHT MARGIN

Function:

The .RIGHT MARGIN command sets the right margin to the position that you specify. This is the position to which a line of text normally extends (compare with .PAGE SIZE). If .JUSTIFY is in effect, the .RIGHT MARGIN value is the position against which text is justified. If .NO JUSTTIFY is in effect, the .RIGHT MARGIN value specifies the maximum length of any text line.

Format:

.RIGHT MARGIN $\begin{bmatrix} +n \\ n \\ -n \end{bmatrix}$

.RM $\begin{bmatrix} +n \\ n \\ -n \end{bmatrix}$

n

Specifies the character position of the new right margin. n must be greater than the value for the left margin. (For example, .RIGHT MARGIN 60 sets the right margin just to the right of the 60th character position.)

+n

Sets the right margin n character positions to the right of the current right margin.

-n

Sets the right margin n character positions to the left of the current right margin.

Characteristics:

- The .RIGHT MARGIN command issues a .BREAK before setting the right margin.

- If you issue .RIGHT MARGIN with a value greater than the page width setting of .PAGE SIZE, the page width expands to the same value as the new right margin. Issuing a .RIGHT MARGIN with a value that is less than the page width has no effect on the page width. To set the page width to its previous value, you must use a .PAGE SIZE command.

Effects of other commands:

- If you issue a .CHAPTER or .APPENDIX command, the .RIGHT MARGIN is set equal to the page width value of .PAGE SIZE.
- If you issue a .PAGE SIZE with a page width less than the current right margin, the right margin is set equal to the new page width.
- .LITERAL issues a .RIGHT MARGIN 150 and .END LITERAL restores the previous right margin setting.

Defaults:

- If you do not issue a .RIGHT MARGIN command, you get .RIGHT MARGIN 60.
- If you issue .RIGHT MARGIN without a value, you get the current page width setting of .PAGE SIZE.

.SEND TOC

Function:

The .SEND TOC command allows you to insert into the table-of-contents intermediate file (the .BTC file) commands, flags, and text that will affect the appearance of your table of contents. See Chapter 4, Creating a Table of Contents, for a more detailed discussion of this command.

Format:

```
.SEND TOC text

.STC text
```

.SET DATE
.SET TIME

.SET DATE and .SET TIME

Function:

The .SET DATE and .SET TIME commands let you specify a date and time to be inserted in your file when you issue the Substitute flag pair, $$, with any of the appropriate date or time words. .SET DATE also sets the date for the .DATE command, which causes the date to appear in running heads. (See .DATE in this chapter and the description of the Substitute flag pair, $$, in Chapter 3.)

Format:

.SET DATE d_1,d_2,d_3	.SET TIME t_1,t_2,t_3
.SDT d_1,d_2,d_3	.STM t_1,t_2,t_3

d_1

A number specifying the day of the month.

d_2

A number specifying the month of the year.

d_3

A number specifying the year (either four digits or the last two digits of the year).

t_1

A number specifying the hour of the day.

t_2

A number specifying minutes past the hour.

t_3

A number specifying seconds past the minute.

If you precede any of these values with a + or -, you will change the corresponding current value by adding to or subtracting from it the value following the + or -.

Characteristics:

- The .SET DATE and .SET TIME commands issue .BREAK commands before doing their main tasks.
- The date or time that you specify or that is in effect by default remains in effect until you issue another .SET DATE or .SET TIME.

Defaults:

- If you do not issue .SET DATE or .SET TIME, issuing a $$time, $$date, or other $$ will give you the date or time that DSR began processing the file.
- If you issue either of these .SET commands, you can retain a previous value by omitting its value from the command. You must, however, type any comma that would have followed it.
- If you issue either command without specifying any values for it, you will get the current date, or the time, as of the second the command is executed.

.SET LEVEL

Function:

The .SET LEVEL command allows you to preset the level of the next section head without issuing a .HEADER LEVEL command (see .HEADER LEVEL).

Format:

$$.\text{SET LEVEL} \begin{Bmatrix} +n \\ n \\ -n \end{Bmatrix}$$

$$.\text{SL} \begin{Bmatrix} +n \\ n \\ -n \end{Bmatrix}$$

n

Specifies the level for the next .HEADER LEVEL command.

+n

Makes the level for the next .HEADER LEVEL command n more than the current level.

-n

Makes the level for the next .HEADER LEVEL command n less than the current level.

Characteristics:

- The .SET LEVEL command can be useful to you if you are using the .REQUIRE command to combine several text files that are to be processed and output as a single file. If you are using some of these text files again, but in different combinations, you can use them without having to modify them and without having to make adjustments to .HEADER LEVEL commands throughout the rest of any document involved. When using .SET LEVEL in .REQUIRE files, note the following:

  1. You should issue all .HEADER LEVEL commands in .REQUIRE files with +n or -n values rather than with n values. Alternatively, you can issue no value with the .HEADER LEVEL command, always specifying the level with .SET LEVEL.

  2. At the end of the .REQUIRE file, you can issue .SET LEVEL with a -n or +n to reset the header level to its value before you entered the current .REQUIRE file.

Example:

Example 2-14 illustrates the use of the .SET LEVEL command. The example file requires another file, called SL3.EXA, which contains the following:

```
.HEADER LEVEL first header in SL3.EXA
Here is text following the first header. Now we will
increment the header level by doing ".SET LEVEL +1":
.SET LEVEL +1
.HEADER LEVEL second header in SL3.EXA
Text following the second header. Now we again
increment the header level:
.SET LEVEL +1
.HEADER LEVEL third header in SL3.EXA
This is the last text in SL3.EXA. At the end of it, we
will set the header level back to where it started by
doing ".SET LEVEL -2".
.SET LEVEL -2
```

Example 2-17: .SET LEVEL

The following shows the input file before it is processed by DSR:

```
One reason for using this command is that it allows
modularization of a .RNO source file or set of Require
files.
.BLANK
Soon we are going to Require a file (SL3.EXA) that has
3 levels of headers. They will appear the first time as
header-levels 1, 2, and 3 (numbered 1.0, 1.1, and
1.1.1). Following that, we will issue a .SET LEVEL +1
command here and then Require our 3-level file again.
This time, the same headers will appear as header-levels
2, 3, and 4 (numbered 1.2, 1.2.1, and 1.2.1.1).
.BLANK
We will also pull in the left margin for SL3.EXA to
make it stand out.
.LEFT MARGIN +3
.REQUIRE 'SL3.EXA'
.BLANK.LEFT MARGIN -3
Now we issue a .SET LEVEL +1 command here at the top
level:
.SET LEVEL +1
Now we Require the file SL3.EXA again:
.LEFT MARGIN +3
.REQUIRE 'SL3.EXA'
.LEFT MARGIN -3
```

The output file is this:

```
     One  reason  for  using  this  command  is  that  it  allows
     modularization  of  an  .RNO  source  file or set of Require
     files.

     Soon we are going to Require a file  (SL3.EXA)  that  has  3
     levels  of  headers.   They  will  appear  the first time as
     header-levels 1, 2, and 3 (numbered 1.0,  1.1,  and  1.1.1).
     Following  that,  we will issue a .SET LEVEL +1 command here
     and then Require our 3-level file  again.   This  time,  the
     same  headers  will  appear  as  header-levels  2,  3, and 4
     (numbered 1.2, 1.2.1, and 1.2.1.1).

     We will also pull in the left margin for SL3.EXA to make  it
     stand out.

        1.0  FIRST HEADER IN SL3.EXA

        Here is text following the first  header.   Now  we  will
        increment the header level by doing ".SET LEVEL +1":

        1.1  Second Header In SL3.EXA

        Text following the second header.  Now we again increment
        the header level:

        1.1.1  Third Header In SL3.EXA - This is the last text in
        SL3.EXA.   At  the end of it we will set the header level
        back to where it started by doing ".SET LEVEL -2."

     Now we issue a .SET LEVEL +1 command here at the top level:
     Now we Require the file SL3.EXA again:

        1.2  First Header in SL3.EXA

        Here is text following the first  header.   Now  we  will
        increment the header level by doing ".SET LEVEL +1":

        1.2.1  Second Header In  SL3.EXA  -  Text  following  the
        second header.  Now we again increment the header level:

        1.2.1.1  Third Header In SL3.EXA - This is the last  text
        in  SL3.EXA.   At  the  end of it, we will set the header
        level back to where it started by doing ".SET LEVEL -2."
```

.SET PARAGRAPH

Function:

The .SET PARAGRAPH command allows you to set values for .PARAGRAPH without issuing a .PARAGRAPH. .SET PARAGRAPH can be especially useful if you plan to issue .AUTOPARAGRAPH or .AUTOTABLE. (See .PARAGRAPH.)

Format:

$$.SET\ PARAGRAPH \left[\begin{bmatrix} n_1 \\ -n_1 \end{bmatrix}, \begin{bmatrix} n_2 \\ -n_2 \end{bmatrix} [,n_3] \right]$$

$$.SPR \left[\begin{bmatrix} n_1 \\ -n_1 \end{bmatrix}, \begin{bmatrix} n_2 \\ -n_2 \end{bmatrix} [,n_3] \right]$$

n_1, n_2, and n_3 are identical to the values of the .PARAGRAPH command:

n_1

(.INDENT) specifies how many character positions to the right of the .LEFT MARGIN setting the first line of text will begin.

$-n_1$

Specifies how many character positions to the left of the .LEFT MARGIN setting the first line of text will begin. $-n_1$ cannot, however, cause the text to begin to the left of character position 0.

n_2

(.SKIP) is the number of blank lines you want inserted before the paragraph. You get additional blank lines if the .SPACING value is greater than 1.

$-n_2$

Specifies that the next line of text be pushed to within n_2 lines of the bottom of the current page by the insertion of blank lines. Every line but the last one retains the line spacing (.SPACING value) that follows it.

n_3

(.TEST PAGE) is the number of lines of text required to be on one page. Unlike the .TEST PAGE command itself, n_3 takes into account any blank lines that .SPACING routinely inserts after each line of text. If there is not enough room on the current page to accommodate that many lines, DSR puts the text on the next page. You can cancel this function by specifying 0 for n_3.

.SKIP

Function:

The .SKIP command inserts a multiple of the number of blank lines that has been specified by the .SPACING command. Contrast this with .BLANK, which inserts only the number of blank lines specified with that command. (See .BLANK.)

Format:

$.SKIP \begin{bmatrix} n \\ -n \end{bmatrix}$

$.S \begin{bmatrix} n \\ -n \end{bmatrix}$

n

The number of .SPACING lines you want to be inserted. For example, if you have specified a .SPACING value of 2 lines and you issue .SKIP without an n value, DSR will insert 2 blank lines (the .SPACING value) by default. If you issue .SKIP 2, DSR will insert 4 blank lines (2 times the .SPACING value), and so on.

-n

Specifies that the next line of text be pushed to within n lines of the bottom of the current page by the insertion of blank lines. Every line but the last one retains the line spacing (.SPACING value) that follows it.

Characteristics:

- The .SKIP command issues a .BREAK before doing its main task.
- .SKIP n does not work at the top of a page, that is, right after a .PAGE or just after .PAGE SIZE length has been exceeded, but .SKIP -n does work under such conditions.
- If there is not enough room on the current page for .SKIP to do exactly as you specified, the .SKIP does as much as it can on that page. It does not finish on the next page.
- If DSR encounters a footnote while executing .SKIP, it considers the line directly above the footnote to be the bottom of the page.

Default:

If you issue no value for .SKIP, you get .SKIP 1.

.SPACING

Function:

The .SPACING command changes the amount of spacing between lines of text.

Format:

```
.SPACING n

.SP n
```

n

The amount of spacing that you want between lines of text. For example, 1 denotes single spacing (no blank lines between lines of text) -- as the text in this manual normally appears. You must specify n, which must be in the range 1 through 5.

Characteristics:

- The .SPACING command issues a .BREAK before doing its main task.
- The .SPACING setting affects the action of .SKIP, .PARAGRAPH, .SET PARAGRAPH, .AUTOPARAGRAPH, and .AUTOTABLE (the test-page values, as well as the skip values, of the last four of these commands).

Default:

If you do not issue a .SPACING command, you get single-spacing (.SPACING 1).

.STYLE HEADERS

.STYLE HEADERS

Function:

The .STYLE HEADERS command changes the formats of the various levels of section heads (.HEADER LEVEL n). Do not confuse the numbers that identify the level of the header (in the range 1 through 6) with the numbers that get printed just to the left of the header title (3.5.2, for example). See .HEADER LEVEL. (See also .NUMBER LEVEL and .DISPLAY LEVEL.)

Header Levels for Documents without Chapters and with Chapters

	Nonchapter Sections	Chapter n Sections
.HEADER LEVEL 1	1.0	n.1
.HEADER LEVEL 2	1.1	n.1.1
.HEADER LEVEL 3	1.1.1	n.1.1.1

Format:

.STYLE HEADERS [n_1],[n_2],...[n_9]

.STHL [n_1],[n_2],...[n_9]

n_1

Specifies the lowest-numbered header level to have a run-in title format. Run-in means that the text immediately follows the header on the same line instead of beginning on a new line. All higher-numbered levels also have run-in formats. If n_1=4, then .HEADER LEVEL 4, 5, and 6 titles are run in to the main text.

n_2

Specifies the highest-numbered header level to have its title printed entirely in uppercase. All lower-numbered levels will also have titles entirely in uppercase. If n_2=4, then .HEADER LEVELs 1, 2, 3, and 4 will have titles in uppercase.

n_3

Specifies the highest-numbered header level to have only the first letter of each word capitalized in the title. All lower-numbered levels will also have titles in mixed format. If n_3=6 all levels will have this case format. All capital letters takes precedence over initial capital letters, if there is a conflict.

n_4

Specifies the lowest-numbered header level to not have a section number to the left of its title. All higher-numbered levels will also not have section numbers to the left of their titles.

n_5

Specifies the lowest-numbered non-run-in header level to have its title centered. All higher-numbered non-run-in levels will also have their titles centered.

n_6

Specifies the number of blank lines you want before section heads.

n_7

Specifies the number of blank lines you want after section heads.

n_8

Specifies the number of lines you want to have available on the current page for the test-page issued by .HEADER LEVEL. Note that n_8 takes into account any blank lines that .SPACING routinely inserts after each line of text (unlike .TEST PAGE). (See also .SKIP.)

n_9

Specifies the number of spaces you want between the section number and the section title. The maximum value is 75.

Characteristics:

- The .STYLE HEADERS command issues a .BREAK before doing its main task.
- You can type 0 or 7 for n_1, n_2, n_3, n_4, or n_5 to affect all levels of headers, even though both 0 and 7 exceed the range of allowable levels.
- In a conflict between n_2 and n_3 caused by an overlapping range, n_2 (all uppercase) takes precedence.

Defaults:

If you do not specify a value for any given n, DSR supplies the following default values:

$n_1=3$	Run-in titles for header levels 3 through 6.
$n_2=1$	Titles in all uppercase for header level 1 only.
$n_3=6$	Titles with only the first letter of every word in uppercase for header level 2 through header level 6.
$n_4=7$	A sequence of numbers (or letters) preceding the section title. (See .DISPLAY LEVELS.)
$n_5=7$	Titles printed starting at the left margin (flush left), not centered.
$n_6=2$	Two blank lines before each header.
$n_7=1$	One blank line after each header.
$n_8=$	Seven more than the test-page value of the most recent .PARAGRAPH or .SET PARAGRAPH command you have issued. If you have not specified such a value, you get 7 plus the .PARAGRAPH default of 2. (See the description of n_8 above for note on .SPACING adjustment.)
$n_9=2$	Two spaces between the section number of the header and the header itself (section title).

.SUBPAGE

.SUBPAGE and .END SUBPAGE

Function:

The .SUBPAGE command begins a new page and a new format of page numbering. It numbers the new page by keeping the previous page number and appending the letter A to it. For example, if the previous page is 10, the first subpage is 10A and the next page becomes 10B unless you have issued an .END SUBPAGE in the meantime. (See also .NUMBER SUBPAGE, .DISPLAY SUBPAGE, .HEADERS ON, .LAYOUT, and .PAGE.)

The .END SUBPAGE command begins a new page and goes back to normal page numbering. If you issued the .END SUBPAGE command on page 2-8D, for example, the new page would be numbered 2-9.

Format:

Enable	Disable
.SUBPAGE	.END SUBPAGE
.SPG	.ES

Characteristics:

- Both the .SUBPAGE and the .END SUBPAGE commands issue .BREAK commands before doing their main tasks.
- You can use the .SUBPAGE command if you are changing parts of a document you are reprinting and you do not want to have to renumber the pages in the rest of a chapter (or, possibly, the rest of the document). You can thus avoid a severe disruption of your index and table of contents.

Default:

.SUBPAGE is not in effect.

.SUBTITLE and .NO SUBTITLE

Function:

The .SUBTITLE command allows you to specify a subtitle for a running head (see .HEADERS ON). This subtitle normally appears on the second line of every page but the first at the leftmost position on a line (character position 0), regardless of the .LEFT MARGIN setting. The .NO SUBTITLE command cancels the .SUBTITLE function. (See also .AUTOSUBTITLE, .TITLE, .FIRST TITLE, and .LAYOUT.)

Format:

Enable	Disable
.SUBTITLE [text]	.NO SUBTITLE
.ST [text]	.NST

text

The title of the running head you want to appear on the second line of the page.

Characteristics:

- The .SUBTITLE command issues a .BREAK before doing its main task.
- A .SUBTITLE command must be the last command on the line.
- You must issue a .SUBTITLE if you want .AUTOSUBTITLE or .DATE to work.

Effects of other commands:

- You can change the position of a running head subtitle on pages by issuing a .LAYOUT command.
- If .AUTOSUBTITLE is in effect, it will override the effects of a .SUBTITLE you have issued. That is, DSR will use .HEADER LEVEL titles for running head subtitles. Any subtitle you have specified in a .SUBTITLE command will be overridden when the first applicable .HEADER LEVEL is encountered.

Defaults:

- If you do not issue either .SUBTITLE or .NO SUBTITLE, you get .NO SUBTITLE.
- If you issue the .SUBTITLE command without specifying subtitle text for it, you will get the effects of .AUTOSUBTITLE (provided you have not issued .NO AUTOSUBTITLE).
- If .AUTOSUBTITLE is in effect, and the text picked up from an applicable .HEADER LEVEL command is wider than the margins that are in effect when the subtitle is displayed, the subtitle is truncated and an ellipsis (...) is appended to the end of it.

.TAB STOPS

.TAB STOPS

Function:

The .TAB STOPS command changes the current positions of tab stops. Each tab character in the input file advances the print carriage to the right to the next tab stop.

Format:

$$\text{.TAB STOPS}\left[\begin{bmatrix}+n_1\\ n_1\\ -n_1\end{bmatrix}, \begin{bmatrix}+n_2\\ n_2\\ -n_2\end{bmatrix}, \ldots \begin{bmatrix}+n_{32}\\ n_{32}\\ -n_{32}\end{bmatrix}\right]$$

$$\text{.TS}\left[\begin{bmatrix}+n_1\\ n_1\\ -n_1\end{bmatrix}, \begin{bmatrix}+n_2\\ n_2\\ -n_2\end{bmatrix}, \ldots \begin{bmatrix}+n_{32}\\ n_{32}\\ -n_{32}\end{bmatrix}\right]$$

$n_1, n_2, \ldots n_{32}$

Character positions that you are defining as new tab stops. They are absolute positions not related to margin settings and start at 0 (the leftmost position). The highest number of tab stops you can have is 32.

n

Specifies the number of the character for a tab stop.

1,2,...32

The 1st,2nd,...32nd tab stops on a line. Tab stops are defined as such by the commas that follow them.

+n

Changes a given position by adding n to its value rather than by directly specifying the value.

-n

Changes a position by subtracting n from its value.

Characteristics:

- If you issue a .TAB STOPS but do not want to change all of the settings, you need not specify those you want to keep. You must, however, retain the commas that would have followed because omissions alter or cancel previous settings. In particular, the previous tab stop settings located after the last number or comma in a .TAB STOPS command are canceled.
- Each .TAB STOPS specification given suspends the use of the previous tab stop settings.
- Current tab stop values must increase from left to right to work properly and each must have a value that is at least two higher than that of the preceding tab stop. Tabs encountered when either of these conditions is not in effect causes DSR to ignore the improper tab setting and bring you to the next legal setting.

- You can delete a tab stop by setting it to a value that is less than the tab stop just preceding the one you are deleting. For example, if you have issued .TAB STOPS 10,15,20,25,30 and you want to delete the tab stop at position 20, you can issue the following:

```
.TAB STOPS ,,12,,
```

- If text overflows past a tab stop, a following tab brings you to the next legal setting.

- If DSR encounters a tab character after passing beyond all tab stops, it treats the tab as a space.

- Tabs work properly inside a .LITERAL block of text.

- .TAB STOPS commands are not affected by .JUSTIFY.

Defaults:

- If you issue a .TAB STOPS command without number or comma specifications, the use of all tab stops is suspended and the tab character is equivalent to a space.

- If you do not issue a .TAB STOPS command, successive tab characters work as if you had set a tab stop every eight positions, that is, as if you had issued .TAB STOPS 9,17,25, and so on. Note, though, that because .PAGE SIZE has a width limit of 150, 149 is the practical limit for .TAB STOPS.

.TEST PAGE

Function:

The .TEST PAGE command allows you to specify an amount of text that you want to appear on a single page in its entirety. If there is not enough room on the current page to accommodate that amount, DSR ends the current page and puts the entire text on the next page.

Format:

```
.TEST PAGE n

.TP n
```

n

The number of lines required to be on one page. This number cannot be omitted and must be positive.

Characteristics:

- The .TEST PAGE command issues a .BREAK before doing its main task.

Effects of other commands:

- The following commands issue .TEST PAGE commands:

  ```
  .FIGURE
  .FIGURE DEFERRED
  .HEADER LEVEL
  .LIST
  .NOTE
  .PARAGRAPH
  .SET PARAGRAPH
  ```

  Unlike .TEST PAGE, however, these commands all take the .SPACING value into account when interpreting n. (See also .SKIP.)

Example:

Example 2-18: .TEST PAGE

Suppose there are exactly four lines left on the current page.

Example A	**Example B**
.TEST PAGE 5	[no .TEST PAGE command]
Line one	Line one
Line two	Line two
Line three	Line three
Line four	Line four
Line five	Line five
produces:	**produces:**
[new page]	Line one
Line one	Line two
Line two	Line three
Line three	Line four
Line four	[new page]
Line five	Line five

.TITLE

.TITLE

Function:

The .TITLE command allows you to specify a title for a running head (see .HEADERS ON). This title normally appears at the top of every page but the first, at the leftmost position on the line (character position 0), regardless of the .LEFT MARGIN setting. (See also .FIRST TITLE, .SUBTITLE, and .LAYOUT.)

Format:

```
.TITLE text

.T text
```

text

The title of the main running head you want to appear.

Characteristics:

- The .TITLE command issues a .BREAK before doing its main task.
- A .TITLE command must be the last command on the line.

Effects of other commands:

- You can change the position of a running head title on pages by issuing a .LAYOUT command.
- If you issue a .CHAPTER (or .APPENDIX) after or instead of a .TITLE, the chapter title becomes the running head title.
- If you are not using .CHAPTER commands in your document, you can have running heads on the first page by issuing a .FIRST TITLE command.

Defaults:

If you do not issue a .TITLE command, you get the title you specified in any .CHAPTER command that is in effect (unless you have issued .NO HEADERS).

.VARIABLE

Function:

The .VARIABLE command finds those sections of your text that contain conditional commands (.IF, .IFNOT, .ELSE, and .ENDIF) and identifies those that will be processed and those that will not ("true" and "false," respectively) by placing identifying characters in the left margin (where change bars would be). You must specify the command line qualifiers /DEBUG or /DEBUG=CONDITIONALS in order to get both the "true" and "false" texts to appear.

Format:

```
.VARIABLE name [y,n]

.VR name [y,n]
```

name

The name you have given to all of the commands in an .IF or .IFNOT block that you want to debug.

y

(Yes) is a single character of your choice that appears in front of lines of text to indicate that they will be processed if you specify /VARIANT instead of /DEBUG in the command line.

n

(No) is a single character of your choice that appears in front of lines of text to indicate that they will not be processed if you specify /VARIANT instead of /DEBUG in the command line.

Characteristics:

- You can issue a .VARIABLE command for as many .IF or .IFNOT blocks as you want, but you should have unique y,n characters for each .VARIABLE unique "name." You can then tell which block it is that will cause a particular portion to be processed or not processed.
- If you issue /VARIANT in the command line, only "true" text will appear.

Example:

Example 2-19 illustrates the use of the .VARIABLE command and also the conditional commands (.IF, .IF NOT, .ELSE, and .ENDIF).

Example 2-19: .VARIABLE

```
.VARIABLE FIXED f,0
.VARIABLE USER u,0
.VARIABLE PASSED p,0
.VARIABLE DIAG d,0

.CENTER;Known Bugs and Deficiencies
.BLANK
.CENTER;PCLS Version V1.1-002

.BLANK 2
This file should be run off with the qualifier /DEBUG in
order to get the benefit of the coded information in the
left-hand margin. The key to the marginal coding is:
.BLANK.LEFT MARGIN 8.TAB STOPS 14
.BREAK;d<TAB>Diagnostic notes.
.BREAK;f<TAB>Fixed bugs -- closed.
.BREAK;p<TAB>Passed to another group  -- closed.
.BREAK;u<TAB>User errors or misunderstandings -- closed.
.LEFT MARGIN 0.BLANK

.LIST
.LEFT MARGIN 6

.IF FIXED
.LIST ELEMENT;[1122] (submitted by Adams) Found in
V1.0-004; VMS only.
.BLANK
PCLS miscounts input records if the /RETRY qualifier is
used.
.BLANK
Fixed in 1.0-005, 10-June-1981.
.ENDIF FIXED
.BLANK
#
.IF USER
.LIST ELEMENT;[1131] (submitted by Clark) Found in
V1.0-006; all operating systems.
.BLANK
Closed: User error.
.BLANK
Reports sometimes print over the line printer page
perforation when the NEWPAGE command is used.
.BLANK
This is not actually a bug. The user wants blank pages but
is neglecting the page-header text in counting lines.
.ENDIF USER
.BLANK
#
.IF PASSED
.LIST ELEMENT;[1149] (submitted by AUSTIN::prine) Found in
V1.0-010; VMS only.
.BLANK
PCLS incurs an access violation upon trying to create a file
in a version-limited directory.
.BLANK
Passed to the I/O group on 6-July-1981.
.BLANK
Fixed in V1.0-011.
.IF DIAG
.BLANK
A problem with the I/O system. It did not properly handle
the message from VMS that notifies the user that the oldest
version was deleted.
.ENDIF DIAG
.ENDIF PASSED
.BLANK
#
.LIST ELEMENT;[1211] (submitted by Clark) Found before
V1.0-012; VMS only.
.BLANK
PCLS access-violates when trying to open the fourth output
file in a user-defined sequence of output files.
.IF DIAG
.BLANK
Problem is likely to be in the EXTOUT module or in something
it calls.
.ENDIF DIAG
.END LIST 0
.BLANK
[End of BUGLIST.RND]
```

The output file is this:

```
                      Known Bugs and Deficiencies

                         PCLS Version V1.1-002


This file should be run off with the qualifier /DEBUG in order to  get
the benefit of the coded information in the left-hand margin.  The key
to the marginal coding is:

          d     Diagnostic notes.
          f     Fixed bugs -- closed.
          p     Passed to another group  -- closed.
          u     User errors or misunderstandings -- closed.

f
f    1.   [1122] (submitted by Adams) Found  in  V1.0-004;   VMS
f         only.
f
f         PCLS miscounts input records if the  /RETRY  qualifier
f         is used.
f
f         Fixed in 1.0-005, 10-June-1981.

u    2.   [1131] (submitted by Clark) Found  in  V1.0-006;   all
u         operating systems.
u
u         Closed:  User error.
u
u         Reports sometimes print  over  the  line printer  page
u         perforation when the NEWPAGE command is used.
u
u         This is not actually a  bug.   The  user  wants  blank
u         pages but is neglecting the page-header text in
u         counting lines.

p
p    3.   [1149] (submitted by AUSTIN::prine) Found in V1.0-010;
p         VMS only.
p
p         PCLS incurs an access violation upon trying to  create
p         a file in a version-limited directory.
p
p         Passed to the I/O group on 6-July-1981.
p
p         Fixed in V1.0-011.
d
d         A problem with the I/O system.  It  did  not  properly
d         handle  the  message  from  VMS that notifies the user
          that the oldest version was deleted.

      4.  [1211] (submitted  by  Clark) Found  before  V1.0-012;
          VMS only.

          PCLS access-violates when trying to  open  the  fourth
          output  file  in  a  user-defined  sequence  of output
          files.
d
d         Problem is likely to be in the  EXTOUT  module  or  in
d         something it calls.

     [End of BUGLIST.RND]
```

.XLOWER
.XUPPER

.XLOWER and .XUPPER

Function:

The .XLOWER command allows you to control the case of index entries specified by .INDEX, .ENTRY, or the Index flag (>). .XUPPER lets DSR control the case of index entries. If .XUPPER is in effect (as it is by default), DSR capitalizes the first character of every index entry, and drops everything else in the entry to lowercase. See Chapter 5, Creating an Index, for a more detailed discussion of these commands.

Format:

```
.XLOWER                  .XUPPER

.XL                      .XU
```

Default:

```
.XUPPER
```

CHAPTER 3

DSR FLAGS AND FLAG CONTROL COMMANDS

DSR flags are special characters that you insert in your text to specify certain characteristics.

Following are descriptions of the operation of all flags available to DSR, along with details of flag recognition.

Table 3-1 lists all the flag names and the characters associated with the flags by default. The table also describes the function of each flag. Flags whose names and characters appear in boldface are enabled by default.

Table 3-1: FLAG DESCRIPTIONS

FLAG NAME	CHAR.	PURPOSE
Accept	**_**	Treat next character as ordinary text
Bold	*	Make next character boldface
Break	|	Allow DSR to break word here if at end of line
Capitalize	<	Capitalize all of the next word
Comment	**!**	Begin a comment
Control	**.**	Start DSR command
Hyphenate	=	Allow hyphenation of word here if at end of line
Index	>	Index the next word
Lowercase	****	Make next character lowercase
Overstrike	%	Overstrike previous character with next character
Period	+	Insert an extra interword space after character
Space	**#**	Insert unexpandable space

(continued on next page)

Table 3-1 (Cont.): FLAG DESCRIPTIONS

FLAG NAME	CHAR.	PURPOSE
Subindex	>	Subindex next word or phrase if in a .INDEX or .ENTRY command
Substitute	$$	Insert date or time
Underline	&	Underline the next character
Uppercase	^	Make next character uppercase

The following flags can be used either singly or paired (with another flag, except as specified):

Accept (can be paired with any flag, including itself)
Bold
Capitalize
Comment
Lowercase
Substitute (can be paired only with itself)
Underline
Uppercase

The following flags are used singly only (except for pairing with the Accept flag):

Break
Hyphenate
Index
Overstrike
Period
Space
Subindex

3.1 FLAG CHARACTER RECOGNITION

You can enable or disable recognition of any flag by issuing a .FLAGS flagname command or a .NO FLAGS flagname command, respectively ("flagname" being the actual name of a flag).

If a flag is not recognized by DSR, it appears as ordinary text, and so does not perform its flag task. If a flag is recognized by DSR, it does not appear in output text. Recognition means only that a flag character will not be taken as text; it does not mean that a flag's function can necessarily be performed. For example, the Underline flag might be recognized, but not cause underlining, because you specified the /NOUNDERLINE qualifier in the DSR command line (see Chapter 6).

You can also suspend recognition of all flags simultaneously (except for Comment and Control) by issuing .NO FLAGS ALL. If you issue this "master switch" command, even those flags with recognition enabled for them by .FLAGS flagname commands will not be recognized. If you later issue .FLAGS ALL, flag recognition will be restored for all flags currently enabled (whether by default or because you have issued the .FLAGS commands for those flags).

Similarly, if a flag is individually disabled (for example, by .NO FLAGS BOLD), the subsequent enabling of all flags (by .FLAGS ALL) will not enable recognition of the individually disabled flag. It is as if you turned off a light switch in a room while the master circuit-breaker was off.

With the .FLAGS flagname command, you can also replace any flag character with another character (that is, with another letter, number, or special character).

For a flag to be recognized, it must be both collectively and individually enabled.

It is possible for you to disable the operation of some flags without affecting their recognition. See .ENABLE/.DISABLE BOLDING, HYPHENATION, INDEXING, OVERSTRIKING, and UNDERLINING.

3.2 REDEFINING A FLAG CHARACTER

Suppose you want to change a flag character from the default to another character. You can redefine a flag character by disabling its current function and then specifying a new character with the appropriate .FLAGS command. You might want to do this if, for example, you are frequently using a default flag character as regular text. By redefining the flag as another character, you avoid having to place an Accept flag before the character each time you use it. (The Accept flag allows flag characters to be accepted by DSR as regular text characters.)

As another example, perhaps you have the chance to move files from a RSTS system, which uses the RNO text-formatting program, to a VAX/VMS system, which uses DSR. In RNO, the ampersand (&) is the bold flag, and the percent sign (%) is the underline flag. The flags cannot be redefined. In DSR, the asterisk (*) is the bold flag and the ampersand is the underline flag, but they CAN be redefined. Therefore, in order to avoid having to change all the ampersands and percent signs in your RNO file to asterisks and ampersands, respectively, you can simply redefine the flags at the beginning of your DSR file as follows:

```
.NO FLAGS UNDERLINE
.FLAG BOLD &
.FLAGS UNDERLINE %
```

Note that, in this example, you must first issue .NO FLAGS UNDERLINE to disable the use of the ampersand, the default flag character for underlining. You must issue a .NO FLAGS flagname command for any flag character that you want to redefine whose operation is currently enabled.

You can also replace a flag character with an ASCII control character, but you must precede it with an Accept flag (_) in the command line. Otherwise, the flag will not be recognized. Note that the operating system may make it difficult for you to input some control characters directly.

In the following example, the Overstrike flag (%) is redefined as the backspace control character (CTRL/H):

```
.FLAGS OVERSTRIKE CTRL/H
```

Here you do not need a .NO FLAGS flagname command because operation of the Overstrike flag is not enabled by default.

3.3 FLAG CHARACTERS

The following is a list of the flag characters grouped according to function. They are listed alphabetically in the next section of the chapter so that you may find their descriptions easily.

Case Flags

Capitalize flag
Lowercase flag
Uppercase flag

Indexing Flags

Index flag
Subindex flag

Spacing Control Flags

Break flag
Hyphenate flag
Period flag
Space flag

Text Emphasis Flags

Bold flag
Overstrike flag
Underline flag

Miscellaneous Flags

Accept flag
Comment flag
Control flag
Sustitute flag pair

Accept Flag

The Accept Flag (_)

The Accept flag causes any character that directly follows it to be accepted as text.

If the character is a punctuation mark after which DSR normally inserts an extra space (for example, a period), you can precede it with the Accept flag to cancel the extra space. You might want to cancel the extra space when you enter terms in which a period does not necessarily end a sentence (for example, Mr. or Mrs.).

If you want to insert a flag character into your text, the easiest way is to precede it with the Accept flag. For example, to insert & (the Underline flag), you would type _&.

For underlining purposes, you can use the flag to cause the acceptance of an expandable space (one you produce by pressing the SPACE bar), because DSR normally does not underline spaces between words.

Default:

Recognition is enabled.

Bold Flag

The Bold Flag (*)

The single character occurrence of the Bold flag causes the next character to be printed in boldface, that is, to be overstruck once. You can cause characters to be overstruck more than once by using the /BOLD qualifier in the DSR command line (see Chapter 6).

Example:

Input:

```
.FLAGS BOLD
Use Types *A and *C
```

Output:

Use Types **A** and **C**

In addition, the operation performed by this flag (as opposed to the flag's recognition) can be enabled or disabled by the .ENABLE BOLDING and .DISABLE BOLDING commands or by the /BOLD and /NOBOLD qualifiers.

You can pair the Bold flag with the Uppercase flag (^*) to turn boldfacing on and pair it with the Lowercase flag (*) to turn boldfacing off. For example:

Input:

```
^*These words are in boldface.\*
```

Output:

These words are in boldface.

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS BOLD command.

The Break Flag (|)

The Break flag tells DSR where it may break a word that occurs at the end of a line. You might want DSR to be able to break a word after a slash (/) or a hyphen (-) that is part of the word (for example, "a yes/no response"). The Break flag allows a line to end where the flag occurs; no hyphen is ever inserted because of it.

If the flag is enabled and inserted at break points, DSR is able to break the word at any of the specified points. If more than one Break flag is present in a word that DSR is breaking at the end of a line, DSR leaves as much of the word as possible on the line. That is, it breaks the word at the last possible Break flag.

The Break flag works the same whether .JUSTIFY or .NO JUSTIFY is in effect.

Example:

Input:

```
.FILL.RIGHT MARGIN 40
.FLAGS BREAK
.BLANK;This is an example of a phrase  (end-|of-|line)  with
break points.
```

Output:

```
This is an example of a phrase (end-of-
line) with break points.
```

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS BREAK command.

Capitalize Flag

The Capitalize Flag (<)

When enabled, the Capitalize flag causes all the letters in the word directly following it to be capitalized with the exception of any letters that may be preceded by an Accept (_) or Lowercase (\) flag.

Capitalization continues until one of the following occurs:

- An expandable space
- A Break flag (|)
- A Hyphenate flag (=)
- Another Capitalize flag
- An Uppercase flag (^)
- A Lowercase flag (\)
- The end of the line

You can pair the Capitalize flag with a circumflex (^) to capitalize all following text up to the next case flag.

Default:

Recognition is disabled. To enable recognition, you must use a .FLAGS CAPITALIZE command.

The Comment Flag (!)

The Comment flag is used to insert comments in the .RNO file. You type the comment text right after the Comment flag. Comments do not appear in the output file.

Example:

```
     .LEFT MARGIN 0.RIGHT MARGIN 60!Place comment here.
```

You can use the flag wherever you use a command, except after a semicolon (;) or at the very beginning of a line (character position 0). Because a semicolon terminates a comment, you cannot include one in your comment. You can, however, include a semicolon in regular text without using an Accept flag.

A semicolon can also be used as a comment flag but only when it is paired with a control flag at the beginning of a line:

```
     .;Comment is here.
```

When not used as a comment flag, a semicolon can be used to terminate a comment or a string of DSR commands. In this case, text that you type after the semicolon will appear in the output file.

You can pair the Comment flag (!) as follows:

- With a dot (.!) to introduce a comment at the beginning of a line:

  ```
  .!Place comment here.
  ```

- With an Accept flag (_!), which allows the character to be taken as ordinary text.

Default:

Recognition is enabled.

Control Flag

The Control Flag (.)

The Control flag is placed at the left margin to begin a string of DSR commands. When you want a dot to be accepted as a text character, you do not need to precede it with an Accept flag (_) as long as the dot is not placed at the left margin. If you do need to have a dot in the 0 character position (and it is not part of a DSR command), you must precede it with an Accept flag. Alternatively, you can use two dots at the beginning of a line; the effect is the same as if you had used an Accept flag. (See also .FLAGS CONTROL.)

Examples:

Input:

```
.INDENT 5
The word "indent" is taken as a command by DSR because it is
preceded by a dot in the left margin.
```

Output:

```
     The word "indent" is taken as a command by DSR  because
it was preceded by a dot in the left margin.
```

(Note that the period at the end of the sentence does not need an Accept flag.)

Input:

```
_.FLAGS BOLD enables recognition of the Bold flag.
```

Output:

```
.FLAGS BOLD enables recognition of the Bold flag.
```

(Here an Accept flag is needed in the input text because the period is placed at the left margin.)

Input:

```
..FLAGS BOLD enables recognition of the Bold flag.
```

Output:

```
.FLAGS BOLD enables recognition of the Bold flag.
```

(Using two dots has the same effect as using the Accept flag.)

Default:

Recognition is enabled.

The Hyphenate Flag (=)

When the Hyphenate flag is enabled and inserted between syllables of a word, DSR knows where the word can be broken at the end of a line. DSR inserts a hyphen where the break occurs.

If DSR does not find it necessary to break the word, however, the hyphen does not appear.

The operation of this flag (as opposed to the flag's recognition) can be enabled or disabled by the .ENABLE HYPHENATION and .DISABLE HYPHENATION commands.

Example:

Input:

```
.FLAGS HYPHENATE
This is an example of a hy=phen=at=ed word.
```

Output:

```
This is an example of a hy-
phenated word.
```

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS HYPHENATE command.

Index Flag

The Index Flag (>)

With the Index flag you can mark words to be indexed in the text of your document. For a more detailed discussion of this flag see Chapter 5, Creating an Index.

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS INDEX command.

The Lowercase Flag (\)

The Lowercase flag causes the letter that directly follows it to appear in lowercase. The flag has no effect if the character following it is not a letter.

The Lowercase flag can be paired as follows:

- With the Underline flag (\&) to stop underlining text.
- With the Bold flag (*) to stop boldfacing characters.
- With itself (\\) to cause the characters following it to be printed in lowercase by default. If you have a file that is in all uppercase, you can put a paired lowercase flag (\\) (or the .LOWER CASE command) at the beginning of the file and then as needed override the temporary lowercase default by using a circumflex (^) to capitalize a letter following it.

Default:

Recognition is enabled.

Overstrike Flag

The Overstrike Flag (%)

When the Overstrike flag is enabled and inserted between two characters, it allows the preceding character to be overstruck by the one following.

This capability allows the printing of characters not normally available, such as a 7 overstruck with a dash (7̶).

Three or more characters can be overstruck, but only if you specify the /BACKSPACE qualifier in the DSR command line. Otherwise only the first and last characters in an overstrike sequence will appear (see Chapter 6).

Example:

Input:

```
.FLAGS OVERSTRIKE
Overstrike this 7%-.
```

Output:

```
Overstrike this 7̶.
```

In addition, the operation performed by this flag (as opposed to the flag's recognition) can be enabled and disabled by the .ENABLE OVERSTRIKING and .DISABLE OVERSTRIKING commands.

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS OVERSTRIKE command.

The Period Flag (+)

DSR routinely inserts an extra expandable space after a period, semicolon, colon, question mark, or exclamation point that is followed by the usual end-of-word space.

The Period flag lets you specify the extra space for other characters.

If the flag is enabled and .FILL is in effect, an extra space occurs when the flag (+) is inserted directly after the character. You must, however, insert the end-of-word space after the flag for it to be effective.

For example, if you have a complete sentence enclosed in quotation marks or parentheses, you may want an extra space after the closing quotation mark or parenthesis. (See also the .PERIOD command in Chapter 2.)

Example:

Input:

```
.FLAGS PERIOD
"What do you mean?"+ There was no response.
```

Output:

```
"What do you mean?"  There was no response.
```

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS PERIOD command.

Space Flag

The Space Flag (#)

The Space flag produces one unexpandable space (not affected by justification) in the output file for every flag character inserted in the input file. If you insert the flag between two words, DSR treats them as one word (though they will appear as separate words in the output file). Therefore, you should not type any spaces before or after typing the Space flag.

The flag can directly follow an Underline flag (&#) to cause the underlining of an unexpandable space.

Default:

Recognition is enabled.

The Subindex Flag (>)

The Subindex flag indicates that the next word or phrase will be indented two characters to the right of the preceding entry. This flag works only if you have issued an .INDEX or .ENTRY command. For a more detailed discussion, see Chapter 5, Creating an Index.

Default:

Recognition is enabled.

Substitute Flag

The Substitute Flag Pair ($$)

This is the only flag that must be paired with itself. When the flag is enabled, it causes either the date or the time to be output, as determined by a name associated with the flag pair. (See also .SET DATE and .SET TIME in Chapter 2.)

When the Substitute flag pair is enabled, any dollar sign character ($), even if it is not paired, must be preceded by an Accept flag if it is to be taken as normal text by DSR.

The following example shows the use of the flag. The output file will contain the date and time that DSR processing of the file began.

Input:

```
.FLAGS SUBSTITUTE
$$Date
$$Time
$$Year
$$Month
$$Day
$$Hours
$$Minutes
$$Seconds
$$Month#$$Day,#$$Year
```

Output:

```
25 Mar 82
10:16:07
1982
March
25
10
16
07
March 25, 1982
```

Default:

Recognition is disabled. To enable recognition, you must use the .FLAGS SUBSTITUTE command.

The Underline Flag (&)

The Underline flag causes the next character to be underlined.

The operation performed by this flag (as opposed to the flag's recognition) can be disabled and reenabled by the .ENABLE UNDERLINING and .DISABLE UNDERLINING commands.

The Underline flag can be paired as follows:

- With the Uppercase flag (^&) to turn underlining on and with the Lowercase flag (\&) to turn underlining off.
- With the Space flag (&#) to cause the underlining of unexpandable spaces.

You can produce underlined text by placing the begin (^&) and end (\&) underline flag pairs before and after the words you want to underline:

Example:

Input:

```
^&To be or not to be\&
```

Output:

<u>To</u> <u>be</u> <u>or</u> <u>not</u> <u>to</u> <u>be</u>

If you want the between-word spaces (either expandable or unexpandable) also to be underlined, you must precede them with Accept flags.

Input:

```
^&To_ be_ or_#not_#to_#be\&
```

Output:

<u>To be or not to be</u>

Default:

Recognition is enabled.

Uppercase Flag

The Uppercase Flag (^)

The Uppercase flag serves the same purpose as a typewriter SHIFT key when you use it just before typing a letter. The flag capitalizes any single letter that directly follows it. It has no effect if the character following it is not a letter.

The Uppercase flag can be paired as follows:

- With a Capitalize flag (^<) to turn on the capitalization of the text that follows (the same as using SHIFT-LOCK on a typewriter).
- With an Underline flag (^&) to turn on underlining of the text that follows.
- With a Bold flag (^*) to turn on boldfacing for the text that follows.
- With itself (^^) if you want to ensure that the case of letters in your input file is maintained in your output file. You can use this flag pair with those commands that control case (such as .HEADER LEVEL or .CHAPTER). When you specify a title, precede it with ^^.

  The operation of the paired Uppercase flag (^^) is similar to that of the .UPPER CASE command, but the .UPPER CASE command does not work within commands as the flag pair does.

Default:

Recognition is enabled.

3.4 FLAG CONTROL COMMANDS

The following is a list of the flag control commands grouped according to function. They are listed alphabetically in the next section of the chapter so that you may find their descriptions easily.

Case Commands

.FLAGS/.NO FLAGS CAPITALIZE
.FLAGS/.NO FLAGS LOWERCASE
.FLAGS/.NO FLAGS UPPERCASE

Indexing Commands

.FLAGS/.NO FLAGS INDEX
.ENABLE/.DISABLE INDEXING
.FLAGS/.NO FLAGS SUBINDEX

Spacing Control Commands

.FLAGS/.NO FLAGS BREAK
.FLAGS/.NO FLAGS HYPHENATE
.ENABLE/.DISABLE HYPHENATION
.FLAGS/.NO FLAGS PERIOD
.FLAGS/.NO FLAGS SPACE

Text Emphasis Commands

.FLAGS/.NO FLAGS BOLD
.ENABLE/.DISABLE BOLDING
.FLAGS/.NO FLAGS OVERSTRIKE
.ENABLE/.DISABLE OVERSTRIKING
.FLAGS/.NO FLAGS UNDERLINE
.ENABLE/.DISABLE UNDERLINING

Miscellaneous Commands

.FLAGS/.NO FLAGS ALL
.FLAGS/.NO FLAGS ACCEPT
.FLAGS/.NO FLAGS COMMENT
.FLAGS/.NO FLAGS CONTROL
.FLAGS/.NO FLAGS SUBSTITUTE

.FLAGS ALL

.FLAGS ALL and .NO FLAGS ALL

The .FLAGS ALL and .NO FLAGS ALL commands function as a master switch for all other .FLAGS/.NO FLAGS command settings, with the exception of the .FLAGS/.NO FLAGS COMMENT and .FLAGS/.NO FLAGS CONTROL commands.

.FLAGS ALL and .NO FLAGS ALL enable and disable recognition of all flags without disturbing other flag command settings. (A useful analogy for flag recognition is turning on a master switch [issuing .FLAGS ALL]--those lights whose switches are in the ON position will go on and those whose switches are in the OFF position will not go on.) See also .ENABLE/.DISABLE BOLDING, HYPHENATION, OVERSTRIKING, and UNDERLINING commands.

Format:

Enable	Disable
.FLAGS ALL	.NO FLAGS ALL
.FLAGS	.NO FLAGS
.FL	.NFL

Default:

.FLAGS ALL

.FLAGS ACCEPT and .NO FLAGS ACCEPT

The .FLAGS ACCEPT and .NO FLAGS ACCEPT commands enable and disable the recognition of the Accept flag.

Format:

Enable	Disable
.FLAGS ACCEPT [k]	.NO FLAGS ACCEPT
.FL ACCEPT [k]	.NFL ACCEPT

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Accept flag character (_) is enabled by default.

.FLAGS BOLD

.FLAGS BOLD and .NO FLAGS BOLD

The .FLAGS BOLD and .NO FLAGS BOLD commands enable and disable the recognition of the Bold flag.

Format:

Enable	Disable
.FLAGS BOLD [k]	.NO FLAGS BOLD
.FL BOLD [k]	.NFL BOLD

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Bold flag character (*) is disabled.

.ENABLE BOLDING and .DISABLE BOLDING

The .ENABLE BOLDING and .DISABLE BOLDING commands enable and disable the boldface function. See the description of the Bold flag in the previous section.

Format:

Enable	Disable
.ENABLE BOLDING	.DISABLE BOLDING
.EBO	.DBO

Default:

Operation of the Bold flag character (*) is enabled, but recognition of the Bold flag is not.

.FLAGS BREAK and .NO FLAGS BREAK

The .FLAGS BREAK and .NO FLAGS BREAK commands enable and disable the recognition of the Break flag.

Format:

Enable	Disable
.FLAGS BREAK [k]	.NO FLAGS BREAK
.FL BREAK [k]	.NFL BREAK

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Break flag character (|) is disabled.

.FLAGS CAPITALIZE

.FLAGS CAPITALIZE and .NO FLAGS CAPITALIZE

The .FLAGS CAPITALIZE and .NO FLAGS CAPITALIZE commands enable and disable the recognition of the Capitalize flag.

Format:

Enable	Disable
.FLAGS CAPITALIZE [k]	.NO FLAGS CAPITALIZE
.FL CAPITALIZE [k]	.NFL CAPITALIZE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Capitalize flag character (<) is disabled.

.FLAGS COMMENT and .NO FLAGS COMMENT

The .FLAGS COMMENT and .NO FLAGS COMMENT commands enable and disable the recognition of the Comment flag.

Format:

Enable	Disable
.FLAGS COMMENT [k]	.NO FLAGS COMMENT
.FL COMMENT [k]	.NFL COMMENT

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Comment flag character (!) is enabled.

.FLAGS CONTROL

.FLAGS CONTROL and .NO FLAGS CONTROL

There is no way to reenable recognition of the Control flag character (the dot that begins a DSR command) if you have issued a .NO FLAGS CONTROL command. You can, however, use .FLAGS CONTROL to change the character that precedes the commands.

Format:

Enable	Disable
.FLAGS CONTROL [k]	.NO FLAGS CONTROL
.FL CONTROL [k]	.NFL CONTROL

k

Specifies a character to replace the current control flag character.

Default:

Recognition of the Control flag character (.) is enabled.

.FLAGS HYPHENATE and .NO FLAGS HYPHENATE

The .FLAGS HYPHENATE and .NO FLAGS HYPHENATE commands enable and disable the recognition of the Hyphenate flag.

Format:

Enable	Disable
.FLAGS HYPHENATE [k]	.NO FLAGS HYPHENATE
.FL HYPHENATE [k]	.NFL HYPHENATE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Hyphenate flag character (=) is disabled.

.ENABLE HYPHENATION and .DISABLE HYPHENATION

The .ENABLE HYPHENATION and .DISABLE HYPHENATION commands enable and disable the hyphenation function.

You can use hyphenation to close up excessive spacing between words, which often occurs when margins are narrow and the line contains several long words.

Format:

Enable	Disable
.ENABLE HYPHENATION	.DISABLE HYPHENATION
.EHY	.DHY

Default:

Operation of the Hyphenate flag character (=) is initially enabled by default, but recognition of the Hyphenate flag is not.

.FLAGS INDEX

.FLAGS INDEX and .NO FLAGS INDEX

These two commands respectively enable and disable DSR recognition of the Index flag.

Format:

Enable	Disable
.FLAGS INDEX [k]	.NO FLAGS INDEX
.FL INDEX [k]	.NFL INDEX

k

Specifies a character to replace the current flag character.

Default:

.NO FLAGS INDEX

.ENABLE INDEXING and .DISABLE INDEXING

These commands enable and disable the operation of the indexing commands, the Index flag, and the /INDEX qualifier.

Format:

Enable	Disable
.ENABLE INDEXING	.DISABLE INDEXING
.EIX	.DIX

Default

.ENABLE INDEXING

See Chapter 5, Creating an Index, for a more detailed discussion of these four commands.

.FLAGS LOWERCASE and .NO FLAGS LOWERCASE

The .FLAGS LOWERCASE and .NO FLAGS LOWERCASE commands enable and disable the recognition of the Lowercase flag.

Format:

Enable	Disable
.FLAGS LOWERCASE [k]	.NO FLAGS LOWERCASE
.FL LOWERCASE [k]	.NFL LOWERCASE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Lowercase flag character (\) is enabled.

.FLAGS OVERSTRIKE and .NO FLAGS OVERSTRIKE

The .FLAGS OVERSTRIKE and .NO FLAGS OVERSTRIKE commands enable and disable the recognition of the Overstrike flag.

Format:

Enable	Disable
.FLAGS OVERSTRIKE [k]	.NO FLAGS OVERSTRIKE
.FL OVERSTRIKE [k]	.NFL OVERSTRIKE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Overstrike flag character (%) is disabled.

.ENABLE OVERSTRIKING and .DISABLE OVERSTRIKING

The .ENABLE OVERSTRIKING and .DISABLE OVERSTRIKING commands enable and disable the overstrike function.

You use the Overstrike flag to create special characters that are not available on the terminal by overstriking any printing character with another. For example, you can overstrike a 7 with a hyphen to create a European 7 (7̶). See the description of the Overstrike flag in the previous section.

Format:

Enable	Disable
.ENABLE OVERSTRIKING	.DISABLE OVERSTRIKING
.EOV	.DOV

Default:

Operation of the Overstrike flag character (%) is initially enabled, but recognition of the Overstrike flag is not.

.FLAGS PERIOD and .NO FLAGS PERIOD

The .FLAGS PERIOD and .NO FLAGS PERIOD commands enable and disable the recognition of the Period flag.

Format:

Enable	Disable
.FLAGS PERIOD [k]	.NO FLAGS PERIOD
.FL PERIOD [k]	.NFL PERIOD

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Period flag character (+) is enabled.

.FLAGS SPACE

.FLAGS SPACE and .NO FLAGS SPACE

The .FLAGS SPACE and .NO FLAGS SPACE commands enable and disable the recognition of the Space flag.

Format:

Enable	Disable
.FLAGS SPACE [k]	.NO FLAGS SPACE
.FL SPACE [k]	.NFL SPACE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Space flag character (#) is enabled.

.FLAGS SUBINDEX and .NO FLAGS SUBINDEX

The .FLAGS SUBINDEX and .NO FLAGS SUBINDEX commands enable and disable DSR's recognition of the Subindex flag (>). If you issue .NO FLAGS SUBINDEX, a right angle bracket (>) will be printed as part of your indexed text instead of causing subindexing. You can also use the .FLAGS SUBINDEX command to change the flag to another character. See Chapter 5, Creating an Index, for a more detailed discussion of this command.

Format:

```
          Enable                   Disable

     .FLAGS SUBINDEX [k]      .NO FLAGS SUBINDEX

     .FL SUBINDEX [k]         .NFL SUBINDEX
```

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Subindex flag character (>) within .INDEX or .ENTRY commands is enabled. The Subindex flag character is always taken as normal text outside of a .INDEX or .ENTRY command.

.FLAGS SUBSTITUTE

.FLAGS SUBSTITUTE and .NO FLAGS SUBSTITUTE

The .FLAGS SUBSTITUTE and .NO FLAGS SUBSTITUTE commands enable and disable the recognition of the Substitute flag pair.

Format:

Enable	Disable
.FLAGS SUBSTITUTE [k]	.NO FLAGS SUBSTITUTE
.FL SUBSTITUTE [k]	.NFL SUBSTITUTE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Substitute flag character ($) is disabled.

.FLAGS UNDERLINE and .NO FLAGS UNDERLINE

The .FLAGS UNDERLINE and .NO FLAGS UNDERLINE commands enable and disable the recognition of the Underline flag.

Format:

Enable	Disable
.FLAGS UNDERLINE [k]	.NO FLAGS UNDERLINE
.FL UNDERLINE [k]	.NFL UNDERLINE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Underline flag character (&) is enabled.

.ENABLE UNDERLINING and .DISABLE UNDERLINING

The .ENABLE UNDERLINING and .DISABLE UNDERLINING commands enable and disable the underline function. You can perform the operation only if both flag recognition and the underline function are enabled. See the description of the Underline flag (&) in the previous section.

Format:

Enable	Disable
.ENABLE UNDERLINING	.DISABLE UNDERLINING
.EUN	.DUL

Default:

.ENABLE UNDERLINING

.FLAGS UPPERCASE

.FLAGS UPPERCASE and .NO FLAGS UPPERCASE

The .FLAGS UPPERCASE and .NO FLAGS UPPERCASE commands enable and disable the recognition of the Uppercase flag (^).

Format:

Enable	Disable
.FLAGS UPPERCASE [k]	.NO FLAGS UPPERCASE
.FL UPPERCASE [k]	.NFL UPPERCASE

k

Specifies a character to replace the current flag character.

Default:

Recognition of the Uppercase flag character (^) is enabled.

CHAPTER 4

CREATING A TABLE OF CONTENTS

TOC is a DSR-related program that generates a table of contents. It creates a table of contents from the titles you have specified by issuing .CHAPTER, .HEADER LEVEL, and .APPENDIX commands, and from formatting information you have specified with .SEND TOC commands.

4.1 USING THE TOC PROGRAM

1. Run DSR specifying the /CONTENTS qualifier in the DSR command line. DSR will process your file and give you a .MEM file as it normally does. In addition, it will give you a file called filnam.BTC, which is in a form acceptable to the TOC program. You can process the input file specifying /NOOUTPUT in addition to /CONTENTS. When you use /NOOUTPUT, DSR will only create a .BTC file.

2. Run TOC. To run TOC, you can either type RUN SYS$SYSTEM:TOC or have a symbol defined in your LOGIN.COM file as follows: TOC :== $TOC. Then type TOC after the dollar sign prompt. TOC will prompt you for the following information (square brackets enclose allowed values for your answers to these questions; parentheses enclose the default answers):

   Specify input file:

   The default file type is .BTC.

   Varying header-level indents? [Y/N] (N)

   A Y response causes each header level to be indented two spaces more than the preceding level, enhancing readability. The default answer is N; headers at second level and below will be indented two spaces more than first-level headers.

   Running page counter? [Y/N] (N)

   A Y response causes running page numbers (1, 2, 3,...) in all table-of-contents entries. Running page numbers are the numbers that appear (enclosed in hyphens) at the bottoms of pages only if you specify .LAYOUT 3,n in the DSR input file. By means of this question, you can specify that running page numbers be used in the table of contents regardless of whether or not you specified .LAYOUT 3,n in the document. The default answer is N.

   Specify deepest header level to include: [n] (99)

   If you just press the RETURN key, titles of section heads of all levels will appear in your table of contents. TOC confirms this default by printing "[All header levels will be included.]". If you specify a number n, you designate the highest-numbered header level whose title will be included in the table of contents. For example, you enter 1 if you want just first-level headers and 2 if you want first- and second-level headers. You enter 0 if you want just chapter and appendix titles (no header levels). If you enter anything other than a number, TOC prints an error message and asks the question again.

Specify deepest header level for which to print trailing dots and page number: [n] (99)

If you just press the RETURN key, all header levels are displayed with trailing dots and page numbers. TOC confirms this default by printing "[Page numbers will be given for all header levels.]". If you specify a number n, you get trailing dots and page numbers for header levels 1 through n. Headers of a higher-numbered level than n are shown without dots or page numbers. Chapter and appendix titles never have trailing dots and page numbers.

Keep chapter/header underlining and bolding? [Y/N] (N)

If you answer Y, any underlining or boldfacing information present in chapter titles, header level titles, or appendix titles is carried over to the table of contents. If you answer with N (the default answer), such emphasis does not appear in the table of contents. (The answer to this question has no effect on any emphasis you may specify with the .SEND TOC command; see Section 4.2.)

Do you want headers numbered? [Y/N] (Y)

If you answer this question with N, the numbers that precede .HEADER LEVEL titles in your document will not appear in your table of contents with their titles. If you answer with Y (the default answer), the numbers will appear along with their titles. Note that headers that appear unnumbered in the document (by use of the .STYLE HEADERS command [see Chapter 2]) are shown in the table of contents WITH numbers.

If you respond to any question simply by pressing RETURN you get the default answer shown in parentheses. If you respond with CTRL/Z to any question after the first one, TOC takes default answers to all further questions. Responding with CTRL/Z to "Specify input file:" causes TOC to quit and return to DCL.

When TOC has processed your file, it gives you a file, called filnam.RNT, that is acceptable to DSR.

3. You can direct DSR to process your .RNT table-of-contents file in either of two ways:

   - Issue a .REQUIRE "filnam.RNT" near the beginning of your .RNO file and rerun the .RNO file through DSR for filnam.MEM, which will contain your finished table of contents.

   - Run filnam.RNT through DSR and get a separate finished table of contents called filnam.MEC.

4.2 TABLE-OF-CONTENTS COMMANDS

This section describes the commands you can use to affect your table-of-contents entries.

.ENABLE TOC and .DISABLE TOC

Function:

The .ENABLE TOC and .DISABLE TOC commands enable and disable the sending of table-of-contents information to the .BTC file. You can issue .DISABLE TOC ... .ENABLE TOC around a block of .HEADER LEVEL commands that you do not want in the table of contents.

Format:

Enable	Disable
.ENABLE TOC	.DISABLE TOC
.ETC	.DTC

Default:

```
.ENABLE TOC
```

.SEND TOC

Function:

The .SEND TOC command lets you specify in your .RNO file the insertion of commands, flags, and text in your .RNT file. This capability gives you more control over the appearance of your finished table of contents. Text sent by .SEND TOC appears without a page number in the finished table of contents.

Format:

```
.SEND TOC [n,]text

.STC [n,]text
```

n

A number reserved for future use by DIGITAL. It should not be used.

text

A command, flag, or ordinary text to be processed as part of the .RNT file by DSR.

Examples:

If you want to insert two blank lines before an item in your table of contents, you can issue the following command in your .RNO file just before the .HEADER LEVEL command you want to affect:

```
.SEND TOC .BLANK 2
```

Your .RNT file will have the .BLANK 2 command in it and your table of contents (in .MEC or as part of .MEM) will have the two blank lines before the entry.

As another example, suppose you have changed the default capitalize flag character to "~". You can make sure this change is reflected in the table of contents as well:

```
     .FLAGS CAPITALIZE ~
     .SEND TOC .FLAGS CAPITALIZE ~
```

Later you can use the flag as follows:

```
     .SEND TOC ^~
     .HEADER LEVEL Sample Header
     .SEND TOC \~
```

CHAPTER 5

CREATING AN INDEX

Two methods are available to you for creating an index:

- Include a /INDEX qualifier in the DSR command line and when DSR is through processing your file, run the TCX program. This is the preferred method.
- Issue a .DO INDEX or .PRINT INDEX command where you want the index to occur (probably at the end of your text file).

5.1 USING THE TCX PROGRAM

The TCX program creates a two-column index with alphabetized entries at the left of each column. Each entry is separated from its page number(s) by a comma. Entries with different first letters are separated by a blank line.

To produce a two-column index, do the following:

1. Generate entries for your index by using the .INDEX and .ENTRY commands or by using the Index flag.
2. Run DSR, specifying /INDEX in the command line. DSR will create two files: the usual formatted text file, called filnam.MEM, and an index file called filenam.BIX, which is in a form acceptable to TCX. You can process the input file specifying /NOOUTPUT in addition to /INDEX. When you use /NOOUTPUT DSR will create only a .BIX file.
3. Run TCX, specifying the .BIX file as the input file. To run TCX, you can either type RUN SYS$SYSTEM TCX or have a symbol defined in your LOGIN.COM file as follows: TCX :== $TCX. Then type TCX after the dollar sign prompt. TCX will prompt you for the following information (square brackets enclose allowed values for your answers to these questions; parentheses enclose the default answers):

   Specify input file:

   The default file type is .BIX.

Additional input? [Y/N] (N)

Each time you respond with Y, TCX repeats "Specify input file:". If you specify more than one input file, they will be merged into a single index. Enter the files in the order in which you would like their corresponding entries to appear in the index.

Running page counters? [Y/N] (N)

A Y response causes running page numbers (1, 2, 3,...) in all index entries. Running page numbers are the numbers that appear (enclosed in hyphens) at the bottoms of pages only if you specify .LAYOUT 3,n in the DSR input file. By means of this question, you can specify that running page numbers be used in the index regardless of whether or not you ever specified .LAYOUT 3,n in the document.

Specify number of index lines per page: [13-80] (55)

You can press the RETURN key for a default value of 55. This value is appropriate for the .PAGE SIZE default for page length (58) and the default arrangement for running heads (.LAYOUT 0).

Specify reserve count for first page: [0-<lines/page-47>] (0)

If you respond by pressing the RETURN key or typing zero, the index is now produced. Otherwise, the number of lines specified is reserved for an index heading, in which case TCX will prompt with:

Specify a single line of input for DSR: [text or .REQUIRE "file-spec"]

Enter a DSR source line (which can be a .REQUIRE command). The text produced by this line is inserted at the top of the first page of the index.

If you respond to any question simply by pressing RETURN, you get the default answer shown in parentheses. If you respond with CTRL/Z to any question after the first one, TCX takes default answers to all further questions. Responding with CTRL/Z to "Specify input file:" causes TCX to quit and return to DCL level.

When TCX finishes the processing, it gives you a file, called filnam.RNX, which is in a form acceptable to DSR. (It also gives you an intermediate file named 001TCX.TMP that contains the entire index before folding; that is, it is a 1-column version of the 2-column index. If you want to do any special processing on the index before final production, such as sorting it according to some special algorithm, then the 001TCX.TMP file is your best starting point.)

4. You can have DSR process your .RNX index file in either of two ways:

   - Issue a .REQUIRE "filnam.RNX" at the end of your .RNO file and rerun the .RNO file through DSR for filnam.MEM, which will contain your finished 2-column index.

   - Run filnam.RNX through DSR and get a separate finished 2-column index called filnam.MEX.

5.2 USING THE .DO INDEX AND .PRINT INDEX COMMANDS

You can use .DO INDEX and .PRINT INDEX to create an index by issuing either command where you want the index to occur (such as at the end of your input file).

Function:

These commands are similar, but unlike the .PRINT INDEX command, .DO INDEX starts a new page before printing the index and prints running head information at the top of the page. Both commands take the entries you specified with .INDEX, .ENTRY, or the Index flag, and print a 1-column index with alphabetized entries on the left. Each entry is separated from page numbers by a dotted line. Entries with different first letters are separated by a blank line.

Format:

```
.DO INDEX [text]          .PRINT INDEX

.DX [text]                .PX
```

text

A title for the index, centered above the first entry.

Characteristics:

- Both commands issue .BREAK commands.
- If you have specified either of these commands in your file and then issue /INDEX in the DSR command line, neither command will generate your index.
- Each time .DO INDEX or .PRINT INDEX is issued, an index is generated containing all indexed items since the last .DO INDEX or .PRINT INDEX (or since the beginning of the file).

Default:

If you do not specify "text" for the .DO INDEX command, the word INDEX appears as the title.

5.3 INDEXING COMMANDS

This section describes the indexing commands that you insert in your input file to mark index entries.

.INDEX

Function:

The .INDEX command lets you specify each word or phrase to be indexed as well as each reference to the word or phrase. (See also .ENTRY and the section describing the Index flag.)

Format:

```
.INDEX topic[>subtopic1...>subtopicn]

.X topic[>subtopic1...>subtopicn]
```

topic[>subtopic$_1$...>subtopic$_n$]

Related words or phrases to be indexed, each subtopic being more specific than the preceding topic or subtopic. Do not include a semicolon (;) anywhere in the .INDEX command line unless you precede it with an Accept flag (_). Otherwise, a semicolon will terminate the command and truncate the topic or subtopic immediately.

>

The Subindex flag, which indicates that the subtopic following it will be indented two characters to the right of the topic or subtopic preceding it. This flag works only within the .INDEX or .ENTRY command. You should not confuse it with the Index flag, which functions in the same way as the .INDEX command.

Characteristics:

- You can use any DSR flags within the .INDEX command to affect the appearance of the index entry.

- Each .INDEX command contributes only one page number, which appears after the last subtopic you specify (or after the topic if you have not specified any subtopics).

  If you issue

  ```
  .INDEX telephones>push-button>black
  ```

  your index will show

  ```
  Telephones
    push-button
      black, 5-1 (or black...........5-1)
  ```

  depending on whether you use the TCX program or use either the .DO INDEX or .PRINT INDEX command. In this example, n is the number of the page on which "black" appears.

  If you later had issued

  ```
  .INDEX Telephones>dial
  ```

  your index would have shown

  ```
  Telephones
    dial, 5-3
    push-button
      black, 5-1
  ```

- If .XUPPER (the default) is in effect, the first letter of "topic" is capitalized and all other letters are printed in lowercase. (See .XLOWER and .XUPPER.)

- Issue the .INDEX command immediately following the occurrence of the topic or last subtopic in the text. In the example above, the .INDEX command should appear right after the word "black" occurs in your text.

- If you specify two topics (or subtopics) that are alphabetically identical but dissimilar with respect to case or flags, separate entries will be generated for them. For example, if .XLOWER is in effect and you issue

  ```
  .INDEX Commands>search
  ```

  on page 1 and

  ```
  .INDEX Commands>SEARCH
  ```

  on page 6,

  your index will show

  ```
  Commands
    SEARCH, 6
    search, 1
  ```

  If you specify "search" in both commands, you get

  ```
  commands
    search, 1, 6
  ```

.ENTRY

Function:

The .ENTRY command is the same as the .INDEX command, except that .ENTRY does not generate a page number.

Format:

.ENTRY topic[>subtopic$_1$...>subtopic$_n$]

.Y topic[>subtopic$_1$...>subtopic$_n$]

topic[>subtopic$_1$...>subtopic$_n$]

Related words or phrases to be indexed, each subtopic being more specific than the preceding topic or subtopic. Do not include a semicolon anywhere in the .ENTRY command line unless you precede it with an Accept flag (_). Otherwise, a semicolon will terminate the command and truncate the topic or subtopic immediately.

>

The Subindex flag, which indicates that the subtopic following it will be indented two characters to the right of the topic or subtopic preceding it. This flag works only within the .INDEX or .ENTRY command. You should not confuse it with the Index flag, which functions like the .INDEX command.

Characteristics:

- You can use .ENTRY to include cross-references in your index. For example, if you issue

  ```
  .ENTRY switches>see qualifiers
  ```

  your index will show

  ```
  Switches
    see qualifiers
  ```

- Because .ENTRY commands do not generate page numbers, you can put them anywhere in your file. It is good practice to collect all such cross-reference .ENTRY commands in one place, perhaps at the beginning of the file.

The Index Flag (>)

With the Index flag, you can mark words to be indexed directly in the text of your document (compare with the .INDEX command).

Characteristics:

- You must enable the Index flag by issuing .FLAGS INDEX. (Do not confuse the Index flag with the Subindex flag (both represented as >); the latter operates only with the .INDEX and .ENTRY commands.)

- Insert an Index flag (>) in your text just before a word you want in your index. All the text following the flag will be a single index item, up to the occurrence of one of the following characters (the flags must be enabled):

  ```
  space
  TAB
  # (Space flag)
  | (Break flag)
  = (Hyphenate flag)
  > (another occurrence of the Index flag)
  end of line
  ```

You can insert more than one word into a single index item even though you are using the Index flag method. To do so, type an Accept flag (_) before each space in the phrase to be indexed.

.ENABLE INDEXING and .DISABLE INDEXING

Function:

These commands enable and disable the operation of the .INDEX and .ENTRY commands, the Index flag, and the /INDEX qualifier. No indexing information is stored or sent to the .BIX file while .DISABLE INDEXING is in effect.

Format:

Enable	Disable
.ENABLE INDEXING	.DISABLE INDEXING
.EIX	.DIX

Default:

.ENABLE INDEXING

.FLAGS INDEX and .NO FLAGS INDEX

Function:

These commands enable and disable DSR recognition of the Index flag.

Format:

Enable	Disable
.FLAGS INDEX [k]	.NO FLAGS INDEX
.FL INDEX [k]	.NFL INDEX

k

A character that redefines the Index flag.

Default:

.NO FLAGS INDEX

.FLAGS SUBINDEX and .NO FLAGS SUBINDEX

Function:

These commands enable and disable, respectively, DSR's recognition of the Subindex flag (>). If you issue .NO FLAGS SUBINDEX, the right angle bracket will be printed as part of your indexed text instead of causing subindexing. You can also use the .FLAGS SUBINDEX command to change the flag to another character.

Format:

Enable	Disable
.FLAGS SUBINDEX	.NO FLAGS SUBINDEX
.FL SUBINDEX	.NFL SUBINDEX

Default:

.FLAGS SUBINDEX

.XUPPER and .XLOWER

Function:

The .XUPPER command lets DSR control the case of index entries specified by .INDEX, .ENTRY, or the Index flag (>). DSR makes the first letter of the topic uppercase and all other letters of the entry lowercase. (See .INDEX.)

The .XLOWER command lets you control the case of index entries. If you have issued .XLOWER, your index entries are accepted in whatever case you type them.

Format:

```
     .XLOWER                    .XUPPER

     .XL                        .XU
```

Default:

     .XUPPER.

5.4 CHECKING YOUR INDEX

The /DEBUG=INDEX Qualifier

You can check to see that your index is in proper order by using the /DEBUG=INDEX qualifier in the DSR command line. (You can type /DEBUG if you do not need to limit debugging just to your index.)

If you issue /DEBUG=INDEX in the DSR command line (or just /DEBUG), all .INDEX and .ENTRY commands will appear in your processed DSR file (filnam.MEM) just where you typed them in your input file (filnam.RNO), except that the case of the letters will always be the same as in the finished index. In addition, if you have enabled index flags, their associated items will show up as if you had specified them with .INDEX commands. See Chapter 6 for a further discussion of the /DEBUG qualifier.

CHAPTER 6

RUNNING DSR

To run DSR, you use the DIGITAL Command Language (DCL) command RUNOFF. The RUNOFF command line consists of the RUNOFF command, input file specifications, and optional qualifiers. Do not confuse the RUNOFF command-line qualifiers with the DSR formatting commands, which are part of your input text file.

6.1 USING FILE TYPES

If the input file type is .RNO, you can omit it in the command line. You may use input file names that do not have the type .RNO, but you must use the complete name and file type. The output file name will be the same as the input file name and the output file type will depend on what the input file type is:

INPUT	OUTPUT
.RNB	.BLB
.RNC	.CCO
.RND	.DOC
.RNE	.ERR
.RNH	.HLP
.RNL	.PLM
.RNM	.MAN
.RNO	.MEM
.RNP	.OPR
.RNS	.STD
.RNT	.MEC
.RNX	.MEX
(none)	.MEM
(other)	.MEM

6.2 RULES FOR COMMAND-LINE QUALIFIERS

Formatting documents by means of the DSR qualifiers allows you to alter the position of the text on all pages of the document, specify effects such as underlining and boldfacing, create an index or table of contents, and otherwise control the appearance of printed output.

You can use the qualifiers to override any conflicting DSR commands or flags included in the input file.

The following rules apply to the use of command-line qualifiers:

- A command line can include as many qualifiers as you want as long as no operational conflicts occur. The command line can be continued on the next line if you specify a hyphen (-) as the last character in the first line.

- You can enter qualifier names in uppercase, lowercase, or mixed case.
- You can enter qualifier names in a truncated form. For example, /BACK is equivalent to /BACKSPACE. VAX/VMS looks only at the first four characters of a command or qualifier. Thus, all commands and qualifiers can be abbreviated to four characters or to the shortest unique abbreviation.
- No spaces are allowed between the qualifier symbol (/) and a qualifier name or between a qualifier name and a numerical or text value (for example, /FORM_SIZE=58).
- The qualifier(s) can follow either the RUNOFF command or an individual file specification. Qualifiers placed after the RUNOFF command will affect all files listed in the command line; qualifiers placed after a file specification will affect only that file. Exceptions are /FORM_SIZE, /SIMULATE, /PAUSE, and /LOG, which affect all files listed in the command line regardless of whether the qualifier is placed after the RUNOFF command or a file specification.

6.3 HOW TO RUN DSR

VAX/VMS command language instructs DSR to access an input file, produce a formatted output file, and direct the output either to the disk (for storage and printing) or to the terminal (for display).

To run DSR, you type the word RUNOFF and the name of the input file after the VAX/VMS dollar sign prompt:

```
$ RUNOFF TEST.RNO
```

You can also just type RUNOFF. In this case, DSR prompts for the file name:

```
_File:
```

If your file type is .RNO (the default file type in DSR), you can omit it when you specify the file name.

6.3.1 Output to Disk

DSR now processes the .RNO file and outputs a .MEM file to the disk.

If no errors are detected, the program returns to the VAX/VMS command level.

```
$ RUNOFF TEST
$
```

If an error is detected, the program responds by printing an error message and a summary of errors when finished. For example:

```
%RUNOFF-W-CJL Can't justify line
on output page 1; on input line 8 of page 1 of file "_DBB5:[GILBERT]
TEST.RNO;1"
DIGITAL Standard Runoff Version V2.0: 1 diagnostic message reported
1 page written to _DBB5:[GILBERT]TEST.MEM;1
$
```

Appendix B contains a list of DSR error messages. You can control the display format of the error messages with the DCL command SET MESSAGE. See the VAX/VMS Command Language User's Guide for further information on SET MESSAGE.

6.3.2 Output to Terminal

If you want to display the processed output file on the terminal (TT:) instead of saving it in a file, type the following:

```
     $ RUNOFF TEST.RNO/OUTPUT=TT:

     (TEST.MEM is displayed on the terminal)

     $
```

6.3.3 Input from Terminal

If you want to insert additional commands before processing, specify the terminal (TT:) as the input device:

```
     $ RUNOFF TT:/OUTPUT=TEST.MEM

     .FLAGS BOLD ~

     .REQUIRE "NEST.RNO"

     ^Z

     $
```

Note that you terminate input from the terminal by typing CTRL/Z (which VAX/VMS displays as ^Z). If errors are detected, the program prints messages as the errors are encountered.

6.3.4 Terminal Input and Output

If you want to test individual DSR commands, flags, or qualifiers by checking them on the terminal, you can specify the terminal as both the input device and the output device. For example:

```
     $ RUNOFF TT:/OUTPUT=TT:

     .FLAGS CAPITALIZE

     <capitalize the first word.

     . RET
```

Display:

```
     CAPITALIZE the first word.
```

To terminate the session, type CTRL/Z.

6.4 COMMAND LINE QUALIFIERS

The following section gives descriptions of DSR command-line qualifiers. The qualifiers are listed in alphabetical order.

/BACKSPACE

The /BACKSPACE qualifier directs DSR to use the Backspace character to produce three special effects:

- Boldfacing characters (see the Bold flag description in Chapter 3) by backspacing and overstriking each flagged character as it is printed.
- Overstriking characters (see the Overstrike flag description in Chapter 3) by backspacing and overstriking each flagged character as it is printed.
- Underlining flagged text (see the Underline flag description in Chapter 3) by backspacing and underlining each flagged character as it is printed. The default underlining character is an underscore (_).

The /BACKSPACE qualifier generally gives more exact underlining and boldfacing for files output on letter-quality printers.

Using /BACKSPACE allows you to overstrike three or more characters by use of the Overstrike flag (see Chapter 3).

If you do not issue the /BACKSPACE qualifier, the printer produces the above effects by issuing a carriage return without a line feed, then printing additional lines that contain only underscores or only the overstruck or bolded text.

Example:

```
$ RUNOFF TEST.RNO/BACKSPACE
```

/BOLD

/BOLD[=number] and /NOBOLD

The /BOLD and /NOBOLD qualifiers enable and disable the boldface function. In addition, a number used with /BOLD specifies the number of times the text is to be overstruck. The /BOLD qualifier does not affect recognition of the Bold flag; it only affects the number of times characters are overstruck.

The default is /BOLD=1. Specifying /BOLD=0 is equivalent to using /NOBOLD. /BOLD=3 gives good results on most line printers.

Examples:

```
     $ RUNOFF MYFILE/BOLD=4

     $ RUNOFF SOURCE.1/BOLD=0

     $ RUNOFF CHAPTER5.V02/NOBOLD
```

/CHANGE_BAR[="c"] and /NOCHANGE_BAR

The /CHANGE_BAR and /NOCHANGE_BAR qualifiers enable and disable the appearance of change bars (|) in the output file.

Using /CHANGE_BAR to enable change bars for an output file is equivalent to entering an .ENABLE BAR command at the beginning of your input file (see .ENABLE BAR in Chapter 2). The /CHANGE_BAR qualifier can also specify a replacement for the change-bar character:

```
     /CHANGE_BAR="x"
```

The specified replacement can be a character that takes up space (such as * or #) or a character that does not take up space (such as CTRL/G). A nonspacing character code can be used:

```
     /CHANGE_BAR=%07/OUTPUT=TT:
```

In this example, the octal code 7 causes the terminal bell to ring every time an altered line of output is encountered.

The change-bar qualifier can be disabled by using /NOCHANGE_BAR.

/NOCHANGE_BAR overrides any .ENABLE BAR command in the file.

Examples:

```
     $ RUNOFF YOURFILE/CHANGE_BAR=%07

     $ RUNOFF TESTS/CHANGE_BAR="*"

     $ RUNOFF A.RNT/NOCHANGE_BAR
```

/CONTENTS

/CONTENTS[=file-spec] and /NOCONTENTS

The /CONTENTS qualifier is used to create a table-of-contents file acceptable to the TOC program. Unless you specify otherwise, the file has a type of .BTC.

You use the .BTC file as input to TOC, which in turn produces an input file (.RNT) that is acceptable to DSR for processing.

The default output file name is the input file name. The default file type is .BTC. The default directory is the input file's directory (unless /OUTPUT has been specified, in which case the .BTC file is written to the output file's directory).

You can use the /CONTENTS qualifier as either a command or a file qualifier. If you specify a /CONTENTS=name qualifier as a file qualifier with an input file specification, the name indicated applies only to that input file. If you specify a /CONTENTS=name qualifier as a command qualifier with the RUNOFF command, the name indicated applies to all input files (except any that have their own /CONTENTS=name qualifiers).

If you omit the /CONTENTS qualifier, the default is /NOCONTENTS.

For further details concerning the creation of a table of contents see Chapter 4.

/DEBUG[=(option[,...])] and /NODEBUG

The /DEBUG qualifier traces the operation of those DSR commands defined by one of the following options by causing the associated commands to appear in the output file:

1. CONDITIONALS

   Specifying CONDITIONALS causes DSR to ignore all conditional commands (.IF, .IFNOT, .ELSE, .ENDIF) in the .RNO file and to include the conditional markers in the .MEM file. (See .VARIABLE in Chapter 2.)

2. FILES

   Specifying FILES causes DSR to print notification of the start of each .REQUIRE in the .MEM file in addition to the text of the .REQUIRE files.

3. INDEX

   Specifying INDEX causes all index items in the .RNO file to be printed in the .MEM file, with each item appearing before the line of text with which it is associated.

   All items that are the result of an .INDEX command or an Index flag are labeled with the word .INDEX, while the results of .ENTRY commands are labeled with the word .ENTRY.

4. CONTENTS

   Specifying CONTENTS causes all .SEND TOC commands in the .RNO file to be printed in the .MEM file.

5. ALL

   Specifying ALL causes all four of the above actions.

If you specify more than one option, separate them with commas and enclose the list in parentheses.

If you do not specify /DEBUG, the default is /NODEBUG. If you specify /DEBUG without a qualifier, the default is /DEBUG=ALL.

Examples:

```
$ RUNOFF HERFILE/DEBUG=CONDITIONALS

$ RUNO A./DEB=FILES

$ RUNOFF/DEBUG=INDEX CHAPTER6

$ RUNOFF SURVEY.DAT/DEBUG=(CONTENTS,INDEX)

$ RUNOFF LIST/DEBUG
```

/DOWN

/DOWN=number

The /DOWN qualifier lets you specify the number of blank lines to be inserted at the top of each page, preceding any header information. The number of blank lines specified affects the maximum number of text lines and header information that can be output on a page. For example, if you issue /DOWN=10 with a .PAGE SIZE of 58 lines, the number of blank lines you want is subtracted from the number of text lines specified. Consequently, each page contains a maximum of 48 lines of text and header information, with 10 blank lines preceding the title line.

If you do not issue /DOWN, no blank lines are inserted except those associated with the print device. If you issue /DOWN with no value you get /DOWN=5.

Examples:

```
     $ RUNOFF HISFILE/DOWN=10

     $ RUNOFF HISTORY./DOWN
```

/FORM_SIZE=number

The /FORM_SIZE qualifier specifies the number of lines that can be accommodated per page of output, including all headers and running feet. The default is 66 lines per page.

DSR normally starts each new page by writing a form-feed character to the output file. If the number of lines on a page exactly equals the form-size, however, DSR assumes that the output device (line printer) will advance to the next page under hardware control. DSR does not write a form-feed in this case, for to do so would leave a blank page.

If you are generating output for a device with other than 66 lines per page, use the /FORM_SIZE qualifier.

/INDEX

/INDEX[=file-spec] and /NOINDEX

The /INDEX qualifier is used to create an index file acceptable to the TCX program. Unless you specify otherwise, the file has a type of .BIX.

You use the .BIX file as input to TCX, which in turn produces an input file (.RNX) that is acceptable to DSR for processing.

The default output file name is the input file name. The default file type is .BIX. The default directory is the input file's directory (unless /OUTPUT has been specified, in which case the .BIX file is written to the output file's directory).

You can use the /INDEX qualifier as either a command or a file qualifier. If you specify a /INDEX=name qualifier as a file qualifier with an input file specification, the name indicated applies only to that input file. If you specify a /INDEX=name qualifier as a command qualifier with the RUNOFF command, the name indicated applies to all input files (except any that have their own /INDEX=name qualifiers).

If you omit the /INDEX qualifier, the default is /NOINDEX.

For further details concerning the creation of an index, see Chapter 5, Creating an Index.

/LOG and /NOLOG

The /LOG and /NOLOG qualifiers allow you to control whether or not DSR writes a termination message to the terminal. The termination message includes the following:

- The version number of DSR
- The number of diagnostic messages reported
- The number of output pages generated
- The output file specification

The default is /NOLOG.

If DSR detects errors in processing a file, it writes the termination message to the terminal.

Examples:

```
     $ RUNOFF PAYROLL/LOG
     DIGITAL Standard Runoff Version V2.0-006: No errors detected
     3 pages written to DBA1:[WHITNEY]PAYROLL.MEM;1
     $

     $ RUNOFF ERRORS.RNO/NOLOG/MESSAGES=USER
     RUNOFF-I-CJL, Can't justify line
     on output page 1; on input line 15 of page 1 of file "_DBA1:[WHITNEY]
     ERRORS.MEM;1"
     DIGITAL Standard Runoff Version V2.0: 1 diagnostic message reported
     3 pages written to DBA1:[WHITNEY]ERRORS.MEM;1
     $
```

/MESSAGES

/MESSAGES=option

The /MESSAGES qualifier lets you specify where you want DSR to display error messages. The options are the following:

OUTPUT	Sends error messages only to the output file
USER	Sends error messages only to the terminal

The default is /MESSAGES=(OUTPUT,USER), which sends messages to the output file and displays them on the terminal. You can prevent error messages from going either to the output file or to the terminal, but you cannot suppress them entirely.

Examples:

```
     $ RUNOFF OURFILE/MESSAGES=OUTPUT

     $ RUNOFF DATA.2/MESSAGES=USER
```

/NONSPACING_UNDERLINE[=%Cn]

The /NONSPACING_UNDERLINE qualifier causes flagged text to be underlined by a nonspacing character such as a bell. The character (%Cn) must be expressed as an octal (%On), decimal (%Dn), or hexadecimal (%Xn) value. The default nonspacing character is the bell (%07).

/OUTPUT

/OUTPUT=file-spec and /NOOUTPUT

The /OUTPUT and /NOOUTPUT qualifiers let you specify where DSR's output should go. The default directory is the user directory, the default file name is the name of the input file, and the default file type depends on the input file type (see Section 6.1).

You can use the /OUTPUT qualifier as either a command or a file qualifier. If you specify /OUTPUT=name as a file qualifier with an input file specification, the name applies only to that input file. If you specify /OUTPUT=name as a command qualifier with the RUNOFF command, the name applies to all input files (except any that have their own /OUTPUT=name qualifiers).

/NOOUTPUT tells DSR not to create an output file. Use /NOOUTPUT with the /CONTENTS and /INDEX qualifiers if you want to generate only .BTC and/or .BIX files (see Chapters 4 and 5). You can also use /NOOUTPUT to check an input file for errors with a minimum of overhead.

Examples:

```
     $ RUNOFF FINAL/OUTPUT=TT:

     $ RUNOFF/OUT=CHI::DBA2:[SULLIVAN]MONTHLY.RPT DRAFT.TXT
```

/PAGES=string

The /PAGES qualifier lets you specify one or more groups of pages to be output; others are suppressed. The string is of the form:

```
     start[:end]
```

or

```
     "start:end[,...],start[:end]"
```

If you omit :end from the last page range, all pages from the start of that page range to the end of the document are output.

Specify multiple page ranges in a quoted string, separating them by commas:

```
     /PAGES="start1:end1,start2:end2,...start5:end5"
```

The maximum number of ranges that you may specify is 5.

If you want only one page, start and end must be the same number:

```
     /PAGE=5-30:5-30
```

Example: To specify output from Chapter 4, page 12, through a single page of the appendix and five pages of the index, type

```
     /PAGES="4-12:A-1,Index-1:Index-5"
```

You must specify page numbers in their default form even if you have a .DISPLAY command in your input file that specifies a different form. For example, for Appendix B, Page B-6, you would specify /PAGE="B-6"; but for Chapter V, Page V-13 (the result of a .DISPLAY command specifying uppercase Roman numerals), you would specify /PAGE="5-13".

You may specify part of a range only if the output you want occurs at the start of an appendix or the index.

For an entire appendix, only the letter is required (for example, /PAGES="A"). For an entire index, only the word "Index" is required (/PAGES="Index").

Examples:

```
     $ RUNOFF ITSFILE/OUTPUT=TT:/PAGE=12

     $ RUNOFF REPORT.TMP/OUTPUT=TT:/PAGE=2-12

     $ RUNOFF DOCUMENT/PAGES="2-9:2-9,A-1:A-5c"
```

/PAUSE

/PAUSE and /NOPAUSE

The /PAUSE qualifier controls whether DSR pauses after printing each page of output. Pausing allows you to insert single sheets of paper or reproduction masters into the output device. This qualifier is intended for use with hardcopy output devices such as "daisy-wheel" printers. Do not use /PAUSE if output for the named device is spooled.

/PAUSE temporarily halts output and the terminal bell rings to remind the operator to insert a new form. Processing resumes after the operator types any character on the keyboard (the character does not print). The default condition is /NOPAUSE.

Example:

```
     $ RUNOFF NEWFILE.RNO/PAUSE/OUT=TT:/FORM_SIZE=60
```

/RIGHT[=number] and /NORIGHT

The /RIGHT qualifier causes the text on each page (including header information) to be shifted to the right by the number of spaces specified. These characters are not deducted from the page width specified in the input file.

If you issue /RIGHT without a value, you get /RIGHT=5. If you issue /RIGHT=0, no shift occurs. If you omit the /RIGHT qualifier, the default is /NORIGHT.

Examples:

```
$ RUNOFF OLDFILE/RIGHT=8

$ RUNOFF X.2/RIGHT
```

/SEPARATE_UNDERLINE

/SEPARATE_UNDERLINE[="c"]

The /SEPARATE_UNDERLINE qualifier causes underlining with separate characters on the next line instead of overprinting with underscores. The character (c) may be expressed as a quoted character or as an octal, decimal, or hexadecimal value. The default separate underlining character is the hyphen (-).

Examples:

```
     $ RUNOFF BLUEFILE/SEPARATE_UNDERLINE

     $ RUNOFF CHAPTER4/SEPARATE_UNDERLINE="*"

     $ RUNOFF CALENDAR.LIS/SEPARATE_UNDERLINE=%075
```

/SEQUENCE and /NOSEQUENCE

The /SEQUENCE qualifier controls whether DSR outputs line numbers from the input file. Line numbers show the input lines that generated each output line to assist debugging. If the input file is not line-sequenced, DSR uses sequential numbering.

The default is /NOSEQUENCE, which produces output without line numbers.

Example:

```
$ RUNOFF NAILFILE.RNO/SEQUENCE
```

/SIMULATE and /NOSIMULATE

The /SIMULATE qualifier controls whether line feeds or form feeds are used to advance to the top of each page. The default is /NOSIMULATE, which uses form feeds.

Normally, DSR skips to the top of a page by means of a form feed. If you use /SIMULATE, DSR does not generate form feeds. Instead, it prints enough blank lines to cause a skip to the top of each new page. /SIMULATE also causes a pause before the first page (but before the first page only, whereas /PAUSE causes a pause before every page). To continue after the pause, type any character. The character is not printed.

You normally use /SIMULATE with hardcopy output devices such as "daisy-wheel" printers.

NOTE

> You can use /SIMULATE only in a command line that you type on a terminal. You cannot use /SIMULATE in a command (.COM) file. Doing so will result in an error.

/UNDERLINE_CHARACTER[="c"] and /NOUNDERLINE

The /UNDERLINE_CHARACTER qualifier allows you to specify the character to be used for normal (overprint) underlining of flagged text. The character (c) may be expressed as a quoted character or as an octal, decimal, or hexadecimal value. The default underlining character is the underscore (_).

The /NOUNDERLINE qualifier allows you to disable all underlining.

Notes:

- You may use more than one underlining qualifier in the command line, but you may specify a character for only one of them.
- You may also use the /BACKSPACE qualifier to specify how underlining is accomplished.

Examples:

```
$ RUNOFF DOCFILE/UNDERLINE_CHARACTER="."

$ RUNOFF DOC.DAT/NOUNDERLINE
```

/VARIANT

/VARIANT=string

The /VARIANT qualifier controls the execution of the conditional commands (.IF, .IFNOT, .ELSE, .ENDIF) by specifying the names of the segments to be processed. (See Chapter 2 for descriptions of the conditional commands.) If you specify multiple names in a string, you must separate them by commas and enclose the string in quotation marks. All variant names must be alphanumeric and must begin with a letter.

Examples:

```
$ RUNOFF LASTFILE/VARIANT=A

$ RUNOFF TEST.TMP/VARIANT=BETA

$ RUNOFF T4.2/VARIANT="A,B,C"
```

APPENDIX A

DSR/RUNOFF COMPATIBLE FEATURES

The features described in this appendix, commonly available with earlier versions of RUNOFF, are supported by DSR only for compatibility with those versions.

A.1 .COMMENT Command

You can use the .COMMENT command to insert in the input file descriptive or explanatory text. Because a semicolon terminates a comment, you cannot include a semicolon in your comment. The comment does not appear in the output file. (See the description of the Comment flag in Chapter 3.)

Format:

```
     .COMMENT text

     .;text
```

A.2 The Endfootnote Flag (!)

In certain versions of RUNOFF, you can terminate a footnote by the insertion of an Endfootnote flag (!). This practice is not recommended. (See .FOOTNOTE and .END FOOTNOTE in Chapter 2.) The flag is recognized by default and, if it is enabled, you can use it only if you have issued a .FOOTNOTE command and have not ended the footnote. You enter the flag at the very beginning of a line or after a semicolon (;) that follows a command.

For example:

```
     !

     or

     .SKIP.LM0;!
```

When paired with the Accept flag (_!), the flag character is taken as ordinary text.

A.3 .FLAGS ENDFOOTNOTE and .NO FLAGS ENDFOOTNOTE Commands

These commands disable and reenable recognition of the Endfootnote flag.

Format:

Enable	Disable
.FLAGS ENDFOOTNOTE [k]	.NO FLAGS ENDFOOTNOTE
.FL ENDFOOTNOTE [k]	.NFL ENDFOOTNOTE

k

A character that can replace the current flag character.

Default:

Recognition is enabled.

A.4 .FLAGS QUOTE and .NO FLAGS QUOTE Commands

These commands are identical in function to .FLAGS ACCEPT and .NO FLAGS ACCEPT. See Chapter 3.

A.5 .HYPHENATION and .NO HYPHENATION Commands

These commands are identical in function to .ENABLE HYPHENATION and .DISABLE HYPHENATION. See Chapter 3.

Format:

Enable	Disable
.HYPHENATION	.NO HYPHENATION
.HY	.NHY

A.6 .LEFT Command

Function:

The .LEFT command is identical in function to .INDENT. See Chapter 2.

Format:

.LEFT $\begin{bmatrix} -n \\ n \end{bmatrix}$

.L $\begin{bmatrix} -n \\ n \end{bmatrix}$

A.7 .LOWER CASE Command

The .LOWER CASE command makes all letters lowercase. This command functions in the same manner as the Lowercase flag pair (\\). See Chapter 3.

Format:

```
.LOWER CASE

.LC
```

A.8 .NUMBER INDEX Command

The .NUMBER INDEX command is intended for use before a .PRINT INDEX command to provide page numbers that are prefixed by the word "Index" followed by a hyphen (-). You should note, though, that the first page of your index will not normally be numbered "Index-1" if you use .PRINT INDEX. Instead, it will be numbered "Index-n", where n is the page on which the .PRINT INDEX command was issued.

Format:

```
.NUMBER INDEX

.NMNDX
```

A.9 .PAPER SIZE Command

The .PAPER SIZE command is the same as the .PAGE SIZE command and can be used interchangeably with it. See Chapter 2.

Format:

.PAPER SIZE n_1,n_2

.PS n_1,n_2

A.10 The Quote Flag (_)

The Quote flag is the same as the Accept flag.

A.11 .STANDARD Command

The .STANDARD command issues several commands to conform to formats of certain DIGITAL documents. Its use is not necessary or recommended.

Characteristics:

- The .STANDARD command issues a .BREAK before doing its other tasks.
- .STANDARD lets you specify a new page width (which you normally do with .PAGE SIZE). If you do not specify a value with .STANDARD, the current page width remains in effect.

- If the page width is either 60 or 70, .STANDARD also sets the page length to 58 or 74, respectively.
- .STANDARD sets the right margin to the value of the page width. (See .PAGE SIZE and .RIGHT MARGIN.)
- The .STANDARD command issues a .JUSTIFY command unless you have issued a .NO AUTOJUSTIFY. (However, note that if .JUSTIFY was in effect before you issued .STANDARD, .NO AUTOJUSTIFY does not cancel it.)
- .STANDARD also issues the commands .PAGING, .LEFT MARGIN 0, .SPACING 1, and .FILL.

Format:

```
            ┌+n┐
.STANDARD   │ n│
            └-n┘

      ┌+n┐
.SD   │ n│
      └-n┘
```

n

The new page width you want.

+n

Sets the page width n character positions to the right of the current page width.

-n

Sets the page width n character positions to the left of the current page width.

Default:

If you do not specify a value for n, the current page width remains in effect.

A.12 .SUBINDEX Command

The .SUBINDEX command and .INDEX command are identical in function.

Format:

```
.SUBINDEX entry

.IX entry
```

Do not precede the entry with a semicolon or include a semicolon in it unless you precede it with the Accept flag (_). If you omit the Accept flag, a semicolon terminates the command and thereby causes the rest of the entry to appear as ordinary text.

A.13 .UPPER CASE Command

The .UPPER CASE command makes your output text the same case as your input text. The command does not translate lowercase input into uppercase output.

Format:

```
.UPPER CASE

.UC
```

The .UPPER CASE command functions in the same manner as the Uppercase flag pair (^^). See Chapter 3.

APPENDIX B

DSR MESSAGES

There are three types of DSR messages:

- Messages that are standard (error/non-error)
- Messages resulting from operator or user error
- Messages resulting from DSR program failures

B.1 FORMAT OF ERROR MESSAGES

The general format of messages displayed by the VAX/VMS operating system is the following:

```
%FACILITY-L-IDENT, text
```

FACILITY The name of a VAX/VMS facility or component (in this case, RUNOFF).

L A severity level indicator. It has one of the following values:

Code	Meaning
S	Success
I	Information
W	Warning
E	Error
F	Fatal error

IDENT A 3-letter abbreviation of the message text. The message descriptions in this appendix are alphabetized by this abbreviation.

text The actual text of the message.

The following section lists the DSR error messages, and gives an explanation and user action for each message. Standard message text, which accompanies error messages, follows in a separate section. Words or characters enclosed in angle brackets (< >) represent character strings, numbers, or file names that appear in the actual message.

B.2 DSR ERROR MESSAGES

%RUNOFF-E-BMS, Bad margin specification: "<string>"

Explanation: An illegal value has been used for a margin setting.

User Action: Correct the margin setting in the input file.

%RUNOFF-E-BVN, Missing or illegal variable name: "<string>"

Explanation: The name specification for a .VARIABLE, .IF, .IFNOT, .ENDIF, or .ELSE command is missing or illegal.

User Action: Supply the missing specification or check to see that it is typed correctly.

%RUNOFF-F-CEM, Comma expected, missing: "<string>"

Explanation: A required comma is missing from the command string.

User Action: Supply the missing comma.

%RUNOFF-W-CJL, Can't justify line

Explanation: The text and spacing are greater than the number of character positions available between the margins.

User Action: Truncate the line or use the Break flag to divide it.

%RUNOFF-W-CNF, Character string expected, not found: "<string>"

Explanation: A required quoted character string is missing from a .REPEAT or a .REQUIRE command.

User Action: Supply the missing string.

%RUNOFF-W-CNS, Command not supported: "<n>"

Explanation: The command is not supported by this version of DSR. n is a number for internal DIGITAL use only. This is a DSR internal error.

User Action: Remove or replace the command.

%RUNOFF-W-COF, Can't open footnote work file "<filename>"

Explanation: DSR cannot create the temporary file required to process the footnote information.

User Action: Check to see whether you have write access to your current default directory. If the problem continues, notify the system manager. This error could also be caused by disk quota problems.

%RUNOFF-F-COI, Can't open input file "<filename>"

> **Explanation:** The input file specification is erroneous, the file does not exist, or you do not have read access to the file.
>
> **User Action:** Check the file specification for errors or check to see whether you have read access to the file. If the problem continues, notify the system manager.

%RUNOFF-F-COO, Can't open output file "<filename>"

> **Explanation:** The output file specification is erroneous or you do not have write access to the file.
>
> **User Action:** Check the file specification for errors or check to see whether you have write access to the file. If the problem continues, notify the system manager.

%RUNOFF-W-COR, Can't open required file "<filename>"

> **Explanation:** The required file specification is erroneous, the file does not exist, or you do not have read access to the file.
>
> **User Action:** Check the file specification for errors or check to see whether you have read access to the file. If the problem continues, notify the system manager.

%RUNOFF-F-COT, Can't open contents file "<filename>"

> **Explanation:** DSR cannot create the .BTC file required to generate a table of contents from the titles and header levels.
>
> **User Action:** Check to see whether you have write access to your current default directory. If the problem continues, notify the system manager.

%RUNOFF-F-COX, Can't open indexing file "<filename>"

> **Explanation:** DSR cannot create the .BIX file required to generate a 2-column index.
>
> **User Action:** Check to see whether you have write access to your current default directory. If the problem continues, notify the system manager.

%RUNOFF-W-CRF, Can't read back saved footnotes

> **Explanation:** DSR has successfully written footnote information to the work file but cannot read it back. There may be one or more 001RNO.TMP files already existing.
>
> **User Action:** Delete any 001RNO.TMP files and reprocess the file. If the problem continues, notify the system manager.

%RUNOFF-F-CRP, Can't recognize page on /PAGES qualifier

Explanation: A page range specification is illegally formatted.

User Action: Correct the page range specification.

%RUNOFF-F-CWF, Can't write footnote file (n)

Explanation: DSR can open the footnote work file but cannot write into it. N is a DSR internal error code.

User Action: Reprocess the file. If the problem continues, notify the system manager.

%RUNOFF-W-DNS, .NO SPACE illegal in this context

Explanation: A .NO SPACE command has been inserted following a command that issues a .BREAK.

User Action: Remove the .NO SPACE command or put it in the correct place.

%RUNOFF-W-DVN, Duplicate variable name: "<string>"

Explanation: An attempt has been made to use a name specification with a .VARIABLE command that has already been used with a previous .VARIABLE, .IF, .IFNOT, or .ELSE command.

User Action: Change the name specification.

%RUNOFF-W-EFD, .END FOOTNOTE doesn't terminate .FOOTNOTE

Explanation: An .ENDFOOTNOTE command is improperly positioned in the file. The command must follow associated footnote data.

User Action: Put the .ENDFOOTNOTE command in the correct place or remove it.

%RUNOFF-W-ELD, .END LITERAL doesn't follow .LITERAL

Explanation: The .END LITERAL command is improperly positioned in the file. The command must follow the .LITERAL command.

User Action: Put the .END LITERAL command in the correct place or remove it.

%RUNOFF-W-EVL, Too many variables: "<string>"

Explanation: The maximum of 20 variables has been exceeded with the inclusion of the specified .VARIABLE or conditional command (.IF, .IFNOT, .ENDIF, or .ELSE). "<string>" is the excess .VARIABLE or conditional command. DSR ignores the command in error.

User Action: Reduce the number of variables.

%RUNOFF-W-FAB, File aborted.

Explanation: DSR has aborted the file for internal reasons. This is a DSR internal error.

User Action: Collect as much information as possible and submit a Software Performance Report.

%RUNOFF-W-FEL, <flagname> flag at end of line ignored

Explanation: This flag cannot occur at the end of a line.

User Action: Remove the flag from the end of the line.

%RUNOFF-W-FNA, Flag not allowed in this context: "<string>"

Explanation: (1) A flag has been used that requires that text directly precede or follow it; (2) a flag character has been replaced by another character (by means of a .FLAGS command) and the previous function of the replacement has not been disabled (by means of a .NO FLAGS command).

User Action: Check your use of the flag and make the appropriate correction.

%RUNOFF-W-FWF, Footnotes won't fit on page

Explanation: There are too many footnotes in the input file or there are too many lines in a footnote.

User Action: Reduce the number or size of the footnotes.

%RUNOFF-I-GFC, Given footnote count incorrect. Correct value is n

Explanation: The actual number of lines generated for the footnote is greater or less than the count specified with the .FOOTNOTE n command.

User Action: None. This is an informational message. If you do not want the message to appear, remove the number from the .FOOTNOTE command.

%RUNOFF-W-IBO, Input buffer overflow: "<string>"

Explanation: Too long a character string has been entered in an internal line buffer. This is a DSR internal error message.

User Action: Collect as much information as possible and submit a Software Performance Report.

%RUNOFF-W-IFT, Illegal in footnote: "<string>"

Explanation: You attempted to use an illegal command between .FOOTNOTE and .ENDFOOTNOTE.

User Action: Remove the command.

%RUNOFF-W-IIF, ^X ignored

Explanation: A control character code (^X) has been illegally entered in the input file and has not been accepted.

User Action: In order to legalize the entry of such codes, you must issue the .CONTROL CHARACTERS command.

%RUNOFF-W-ILC, Illegal command: "<string>"

Explanation: This command is illegal. Command or values you specify for it are unrecognizable.

User Action: Check for spelling errors in the command keyword(s).

%RUNOFF-E-ILE, Internal logic error (<string>)

Explanation: DSR has detected an internal failure at the location specified by the indicated string. This is a DSR internal error.

User Action: Collect as much information as possible and submit a Software Performance Report.

%RUNOFF-W-INI, Improperly nested: "<string>"

Explanation: Your conditional commands (.IF/.IFNOT and .ELSE/.ENDIF) are improperly matched.

User Action: Correct the commands so that they match.

%RUNOFF-W-INM, Illegal number value: "<string>"

Explanation: The argument value is too large, too small, illegal, or not a number.

User Action: Check to see that the value is formatted properly.

%RUNOFF-W-ITC, Index entry too complicated or long

Explanation: The entry exceeds the limit of the index buffer for a single item.

User Action: Reduce the length of the index entry.

%RUNOFF-W-ITD, .IF commands nested too deeply: "<string>"

Explanation: The number of nested conditional commands (.IF, .IFNOT) exceeds the maximum limit of 10.

User Action: Reduce the number of conditional commands.

%RUNOFF-F-IVS, Illegal /VARIANT qualifier

Explanation: The name specification for conditional commands is missing or illegal.

User Action: Correct the name specification.

%RUNOFF-W-JEC, Unexpected text at end of command: "<string>"

Explanation: Extraneous text (not a valid terminator) follows this command.

User Action: Insert a valid terminator after the command.

%RUNOFF-W-LDE, Literal doesn't end with .END LITERAL: "<string>"

Explanation: An .END LITERAL command does not immediately follow the final line of a .LITERAL n command (where n is the number of lines to be produced).

User Action: Put the .LITERAL command on the right line or remove the n value.

%RUNOFF-W-LTC, Line too complicated: "<string>"

Explanation: There is excessive use of boldfacing, underlining, overstriking, and hyphenation on the line.

User Action: Simplify the line.

%RUNOFF-W-MEI, Missing at least one .ENDIF command

Explanation: End-of-file has been detected and at least one conditional command (.IF/.IFNOT) is not legally terminated.

User Action: Insert the missing conditional command(s). Check to see that .IF/.ENDIF commands are matched within .REQUIRE files.

%RUNOFF-W-MFN, Number illegal or ill-formed: "<string>"

Explanation: A plus sign (+) or a minus sign (-) has been used alone instead of in +n or -n.

User Action: Insert the n value.

%RUNOFF-W-MFS, Missing or improperly delimited file spec: "<string>"

Explanation: The file specification for the .REQUIRE command is not enclosed in apostrophes or quotation marks.

User Action: Enclose the file specification in apostrophes or quotation marks.

%RUNOFF-W-MNL, Missing number or letter: "<string>"

Explanation: A required numerical or text value is missing.

User Action: Insert the missing value.

%RUNOFF-W-MQS, Missing or improperly delimited quoted string: "<string>"

Explanation: A beginning or ending apostrophe or quotation mark is missing from the quoted text value.

User Action: Insert the missing apostrophe or quotation mark.

%RUNOFF-W-MRC, Another n crossed margin or bad right indent attempt(s) detected and accumulated. Now being reported

Explanation: A specific number (n) of similar horizontal spacing errors have been detected and are being reported. These errors are triggered by .CHAPTER or .APPENDIX commands or by the end-of-file. The count of errors is cleared after each .CHAPTER and .APPENDIX command, or after the end-of-file.

User Action: Check the margin and indentation settings.

%RUNOFF-W-MRG, Margins crossed, or attempted indent too far right

Explanation: The right margin has crossed the left margin or the left margin has crossed the right margin, or an illegal inaent has been specified.

User Action: Check the margin and indentation settings.

%RUNOFF-W-NIA, Negative indent attempted

Explanation: You attempted to indent text to the left of the left margin limit of 0.

User Action: Correct the indentation or left margin setting.

%RUNOFF-W-NIC, Another n negative indent(s) detected and accumulated. Now being reported

Explanation: A specific number (n) of negative indentations (to the left of .LM0) have been detected and are being reported. These errors are triggered by .CHAPTER or .APPENDIX commands or by the end-of-file. The count of errors is cleared after each .CHAPTER and .APPENDIX command or after the end-of-file.

User Action: Correct the indentation or left margin setting.

%RUNOFF-W-NSF, .END LIST/.END NOTE not in same file as .LIST/.NOTE: "<string>"

Explanation: You made an illegal attempt to start a list or note in one file and end it in another.

User Action: End the list or note in the same file in which it was begun.

%RUNOFF-W-NTD, Files nested too deep: "<string>"

Explanation: You attempted to nest .REQUIRE files more than 10 deep.

User Action: Reduce the number of nested .REQUIRE files.

%RUNOFF-W-PWF, Page number won't fit on title

Explanation: The title (on the first header information line) extends into the page number field. Therefore, the page number will not fit.

User Action: Reduce the length of the title.

%RUNOFF-W-QST, Quoted string too long: "<string>"

Explanation: A quoted text value contains too many characters.

User Action: Reduce the number of characters. In some cases, only one quoted character is allowed (for example, the .DISPLAY ELEMENTS command). In other cases, up to 150 characters are allowed (for example, the .REPEAT command).

%RUNOFF-W-RTL, Input record too long: truncated "<string>"

Explanation: Too many characters were entered on a single input line. The excess characters have been discarded.

User Action: Divide the record into several smaller ones.

%RUNOFF-W-SKC, .ENDIF/.ELSE not in same file as .IF/.IFNOT: "<string>"

Explanation: You attempted to use associated conditional commands in separate files.

User Action: Put the associated conditional commands in the same file.

%RUNOFF-W-SSR, Restricted character "^\" (octal 34) encountered in text, ignored

Explanation: DSR encountered the restricted character (octal 34) in the input file.

User Action: Remove the restricted character. You cannot use it in a DSR input file.

%RUNOFF-W-STD, Too many nested .NOTEs and/or .LISTs: "<string>"

Explanation: You attempted to nest more than 20 notes or 20 lists (or more than 7 within a footnote).

User Action: Reduce the number of nested notes or lists.

%RUNOFF-W-TAR, No text allowed after .REQUIRE command: "<string>"

Explanation: You attempted to insert commands or text immediately after a .REQUIRE command file specification.

User Action: Put the commands on a separate line.

%RUNOFF-W-TFE, Too few end commands

Explanation: An .END command (for example, .END LIST) is missing.

User Action: Insert the missing .END command.

%RUNOFF-W-TMF, Too many footnotes, excess footnotes merged

Explanation: You have more than 20 footnotes in a file. All footnotes after the 20th are merged into one large footnote and output after the next page of text. (This large footnote will probably exceed the page length.)

User Action: Use fewer footnotes, shorten them, or spread them out.

%RUNOFF-F-TMP, Too many page ranges on /PAGES qualifier

Explanation: More than five separate ranges have been specified with the qualifier.

User Action: Reduce the number of ranges specified.

%RUNOFF-W-TMT, Too many tab settings; excess ignored: "<string>"

Explanation: More than 40 separate settings have been used with a single .TAB STOPS command.

User Action: Reduce the number of tab settings.

%RUNOFF-F-TMV, Too many /VARIANTs

Explanation: More than 20 variable names have been used with the qualifier.

User Action: Reduce the number of variable names.

%RUNOFF-W-TTL, Text too long: "<string>"

Explanation: The text value for the command (for example, .TITLE, .CHAPTER, or .NOTE) is too long for the specified margin setting or page size, or it contains excessive use of underlining, boldface, or overstriking.

User Action: Reduce the length of, or simplify, the text.

%RUNOFF-W-UDS, Undefined symbol: "<string>"

Explanation: DSR has detected an unrecognizable symbol, for example, $$name, where name has not been defined.

User Action: Remove the symbol.

%RUNOFF-W-UME, Unmatched end command: "<string>"

Explanation: An .END command (for example, .END NOTE) has been detected, but the associated beginning command (.NOTE) has not.

User Action: Insert the associated beginning command.

%RUNOFF-W-URE, Unrecoverable error processing record <ln> on page <pn> of input file "<filename>"

Explanation: An unrecoverable I/O error has occurred, terminating the processing at the specified edit line (ln) and page number (pn) of the input file. This is a DSR internal error.

User Action: Collect as much information as possible and submit a Software Performance Report.

%RUNOFF-W-URR, Unrecognized request: "<n>"

Explanation: DSR detected an unrecognizable request. This is a DSR internal error.

User Action: Try reprocessing the file, or collect as much information as possible and submit a Software Performance Report.

%RUNOFF-W-XFL, Index overflow, results undefined

Explanation: During the construction of a 1-column index, the number of entries exceeded available memory. The excess entries were discarded. This is a DSR internal error.

User Action: Collect as much information as possible and submit a Software Performance Report.

B.3 DSR STANDARD MESSAGE TEXT

The following is standard message text, some of which accompanies only error messages.

See command on input line <ln> of page <pn> of file "<filename>"

> This is a standard error message line that specifies the location of an error-related command in the input file.
>
> The message indicates the name of the input file (filename) in which the command is located and the file page number (pn) and line number (ln) where the error occurs.

on output page pn; on input line <ln> of page <pn> of file "<filename>"

> This is a standard error message line that specifies the location of an error string.
>
> The message indicates the following: the number of the page (pn) in the output file affected by the error, the name of the input file (filename), and the input page (pn) and line number (ln) in which the error occurred.

DIGITAL Standard Runoff Version V2.0-0XX: n diagnostic message(s) reported

> This is a standard error message line that specifies the actual number of error messages output.

DIGITAL Standard Runoff Version V2.0-0XX: No errors detected

> This message indicates the successful processing of a file by DSR.

page(s) written to "<filename>"

> This message indicates the number of pages written to the specified output file (filename).

APPENDIX C

DSR COMMAND SUMMARY

All DSR commands are listed here for the sole purpose of indicating acceptable formats. Each command is shown in its shortest possible form and its full form. The full form also indicates parameter(s) and punctuation. See the descriptions of individual commands for detailed explanations.

Format:

bolded commands are defaults		()	Delimit parameter défaults
{}	Delimit comments	{""}	Delimit alternate forms
$Y \longrightarrow$	Decimal Octal Hexadecimal Roman uppercase Roman lowercase Roman mixed case Letters uppercase Letters lowercase Letters mixed case	$C \longrightarrow$ $\# \longrightarrow$	A character A number

The following is a list of all DSR commands. The commands are listed in alphabetical order by the full name of the command, not by the abbreviation.

.AX	.APPENDIX [text]		
.AJ	**.AUTOJUSTIFY**		
.AP	.AUTOPARAGRAPH {cancels .AUTOTABLE}		
.AST	**.AUTOSUBTITLE** [#] (1) {enabled by .SUBTITLE}		
.AT	.AUTOTABLE {cancels .AUTOPARAGRAPH}		
.BB	.BEGIN BAR		
.B	.BLANK [#]		
.BR	.BREAK	{"." or ".<space>"}	
.C	.CENTER [#][;text] {".CENTRE"}		
.CH	.CHAPTER [text]		
.!	.COMMENT [text]	{".;"}	{';' not allowed in text}
.CC	.CONTROL CHARACTERS		
.D	.DATE		
.DBB	**.DISABLE BAR**		
.DBO	.DISABLE BOLDING		
.DHY	.DISABLE HYPHENATION		
.DIX	.DISABLE INDEXING		
.DOV	.DISABLE OVERSTRIKING		
.DTC	.DISABLE TOC		
.DUL	.DISABLE UNDERLINING		

.DAX	.DISPLAY APPENDIX Y (lu)
.DCH	.DISPLAY CHAPTER Y (d)
.DLE	.DISPLAY ELEMENTS 'C',Y,'C' ('',d,'')
.DHL	.DISPLAY LEVELS Y,Y,Y,Y,Y,Y (d,d,d,d,d,d)
.DNM	.DISPLAY NUMBER Y (d)
.DSP	.DISPLAY SUBPAGE Y (lu)
.DX	.DO INDEX [text] (if no text, the word INDEX is used)
.ELSE	.ELSE name
.EBB	.ENABLE BAR
.EBO	.ENABLE BOLDING
.EHY	.ENABLE HYPHENATION
.EIX	.ENABLE INDEXING
.EOV	.ENABLE OVERSTRIKING
.ETC	.ENABLE TOC
.EUL	.ENABLE UNDERLINING
.EI	.ENDIF name
.EB	.END BAR
.EFN	.END FOOTNOTE
.ELS	.END LIST [#]
.EL	.END LITERAL
.EN	.END NOTE [#]
.ES	.END SUBPAGE
.Y	.ENTRY TEXT
.FG	.FIGURE [#]
.FGD	.FIGURE DEFERRED [#]
.F	.FILL
.FT	.FIRST TITLE
.FL AC	.FLAGS ACCEPT [C] (_) {".FLAGS QUOTE"}
.FL	.FLAGS [ALL]
.FL BO	.FLAGS BOLD [C] (*)
.FL BR	.FLAGS BREAK [C] (|)
.FL CA	.FLAGS CAPITALIZE [C] (<)
.FL COM	.FLAGS COMMENT [C] (!)
.FL CON	.FLAGS CONTROL [C] (.)
.FL E	.FLAGS ENDFOOTNOTE [C] (!)
.FL H	.FLAGS HYPHENATE [C] (=) {".HY" and ".HYPHENATION"}
.FL I	.FLAGS INDEX [C] (>)
.FL L	.FLAGS LOWERCASE [C] (\)
.FL O	.FLAGS OVERSTRIKE [C] (%)
.FL P	.FLAGS PERIOD [C] (+)
.FL SP	.FLAGS SPACE [C] (#)
.FL SUBI	.FLAGS SUBINDEX [C] (>)
.FL SUBS	.FLAGS SUBSTITUTE [C] ($)
.FL UN	.FLAGS UNDERLINE [C] (&)
.FL UP	.FLAGS UPPERCASE [C] (^)
.FN	.FOOTNOTE [#]
.HL	.HEADER LEVEL [#] text {';' not allowed in text}
.HD LO	.HEADERS LOWER
.HD M	.HEADERS MIXED
.HD	.HEADERS [ON]
.HD U	.HEADERS UPPER
.IF	.IF name
.IN	.IFNOT name
.I	.INDENT [#] (paragraph indent) {".L" or ".LEFT"}
.X	.INDEX text {';' not allowed in text} {".IX" or ".SUBINDEX"}
.j	.JUSTIFY
.K	.KEEP
.LO	.LAYOUT [#,#]
.LM	.LEFT MARGIN [#] (0)
.LS	.LIST [#][,"C"]
.LE	.LIST ELEMENT
.LT	.LITERAL
.LC	.LOWER CASE
.NAJ	.NO AUTOJUSTIFY

.NAP	**.NO AUTOPARAGRAPH** {cancels .AUTOTABLE}
.NAST	.NO AUTOSUBTITLE
.NAT	**.NO AUTOTABLE** {cancels .AUTOPARAGRAPH}
.NCC	**.NO CONTROL CHARACTERS**
.ND	**.NO DATE**
.NF	.NO FILL
.NFL AC	.NO FLAGS ACCEPT {".NO FLAGS QUOTE"}
.NFL	.NO FLAGS [ALL]
.NFL BO	**.NO FLAGS BOLD**
.NFL BR	**.NO FLAGS BREAK**
.NFL CA	**.NO FLAGS CAPITALIZE**
.NFL COM	.No FLAGS COMMENT
.NFL CON	.NO FLAGS CONTROL
.NFL E	.NO FLAGS ENDFOOTNOTE
.NFL H	**.NO FLAGS HYPHENATE**
.NFL I.	**.NO FLAGS INDEX**
.NFL L	.NO FLAGS LOWERCASE
.NFL O.	**.NO FLAGS OVERSTRIKE**
.NFL P	**.NO FLAGS PERIOD**
.NFL SP	.NO FLAGS SPACE
.NFL SUBI	.NO FLAGS SUBINDEX
.NFL SUBS	**.NO FLAGS SUBSTITUTE**
.NFL UN	.NO FLAGS UNDERLINE
.NFL UP	.NO FLAGS UPPERCASE
.NHD	.NO HEADERS
.NJ	.NO JUSTIFY
.NK	**.NO KEEP**
.NNM	.NO NUMBER
.NPA	.NO PAGING
.NPR	.NO PERIOD
.NSP	.NO SPACE
.NST	**.NO SUBTITLE**
.NT	.NOTE [text] (The word "NOTE" centered above body of note)
.NMAX	.NUMBER APPENDIX #|C[C[C[C[C]]]] (+1)
.NMCH	.NUMBER CHAPTER # (+1)
.NMNDX	.NUMBER INDEX {NOT recommended}
.NMLV	.NUMBER LEVEL [#],[#],[#],[#],[#],[#]
.NMLS	.NUMBER LIST [,]# (1 {or as per .DISPLAY ELEMENTS})
.NMPG	**.NUMBER [PAGE]** [#] (+1)
.NMR	.NUMBER RUNNING #
.NMSPG	.NUMBER SUBPAGE #
.PG	.PAGE
.PS	.PAGE SIZE [#[,#]] {".PAPER SIZE"} (58,60) {length,width}
.PA	**.PAGING**
.P	.PARAGRAPH [#],[#],[#] (5,1,2 {or prev .P or .SET PARA})
.PR	**.PERIOD**
.PX	.PRINT INDEX
.RPT	.REPEAT #,'text' {max string length is 50 char's}
.REQ	.REQUIRE 'filename.typ'
.R	.RIGHT [#];text
.RM	.RIGHT MARGIN [#] (page width)
.STC	.SEND TOC #,text
.SDT	.SET DATE [#],[#],[#] (current date) {.FL SUBS required}
.SL	.SET LEVEL # {used by .HEADER LEVEL}
.SPR	.SET PARAGRAPH [#],[#],[#]
.STM	.SET TIME [#],[#],[#] (current time) {.FL SUBS required}
.S	.SKIP # (1) {tied to .SPACING command}
.SP	.SPACING # (1)
.SD	.STANDARD # (current page width) {.RM.J.PA.LM0.SP.F} {NOT recommended}
.STHL	.STYLE HEADERS [#1],[#2],...,[#9] (3,1,6,7,7,2,1,+7,2)
.SPG	.SUBPAGE
.ST	.SUBTITLE [text] (text will come from .HL) {".SUBTTL"}
.TS	.TAB STOPS #[,#[,#...]] (8,16,24,32...256)

.TP	.TEST PAGE #
.T	.TITLE text (text from previous .CHAPTER command)
.UC	.UPPER CASE {allows O/P as I/P}
.VR	.VARIABLE name C,C {used with conditional commands}
.XL	.XLOWER
.XU	.XUPPER

APPENDIX D

TEMPLATE SAMPLES

D.1 MEMO TEMPLATE

The following provides an example of a memorandum, and the manner in which it was derived.

```
+-------------+
| | | | | | | |
|d|i|g|i|t|a|l|      I N T E R O F F I C E   M E M O R A N D U M
| | | | | | | |
+-------------+
```

```
To: R. Marschall                        Date: 25 March 1982
                                        From: R. Freitag
                                        Dept: Software Engineering
                                        Ext:  1234
                                        Loc/Mail Stop: XYZ/J14
                                        File: MEMO.RNO


Subject:  DSR Memo Template

                  (The text is input here.)
```

Memo Template Formatting

The following shows the formatting of the memo template.

```
.ps60,70
.flags bold
.flags substitute
.b
.nf
^*+-------------+
| | | | | | | |
|d|i|g|i|t|a|l|       I N T E R O F F I C E    M E M O R A N D U M
| | | | | | | |
+-------------+\*
.ts39,53.lm39
.b2.i-39
To: R.#Marschall                       Date: $$day $$month $$year
From: R.#Freitag
Dept: Software Engineering
Ext:##1234
Loc/Mail Stop: XYZ/J14
File: MEMO.RNO
.ts20.lm0.b3
Subject:  DSR Memo Template
.b2
.f.j
.c;(The text is input here.)
```

D.2 REPORT TEMPLATE

The following provides an example of a report and the manner in which it was derived.

```
+-------------+
| | | | | | | |
|d|i|g|i|t|a|l|      I N T E R O F F I C E    M E M O R A N D U M
| | | | | | | |
+-------------+

To:  List                    Date:  25 March 1982
                             From:  A. S. Sullivan
                             Dept:  Software
                             Ext.:  1000
                             Loc.:  ABC-4/Q21
                             File:  <MMMYY>.RNO
                             Memo:  not used     Rev.:  0
 List: W. S. Gilbert
       R. Carte
       G. Shaw
 cc:

Subj.:  Report for March, 1982


1.0  ACCOMPLISHMENTS

2.0  NOT ACCOMPLISHED

3.0  SCHEDULED FOR NEXT MONTH

4.0  PROBLEMS

5.0  SOLUTIONS

6.0  MISCELLANEOUS


[end of<MMMYY>.RNO]
```

Report Template Format

The following shows the formatting of the report template.

```
.flags substitute
.b2.lm0.nf.ts13,25,38,50,63

+-------------+
| | | | | | | |
|d|i|g|i|t|a|l|       I N T E R O F F I C E   M E M O R A N D U M
| | | | | | | |
+-------------+
.b2.ts30
To:##List                       Date:##$$day $$month $$year
                                From:##A.#S.#Sullivan
                                Dept:##Software
                                Ext.:##1000
                                Loc.:##ABC-4/Q21
                                File:##<MMMYY>.RNO
                                Memo:##not used#####Rev.:##0
.lm 7.i-6
List: W. S. Gilbert
R. Carte
G. Shaw
.lm 5.i-4
cc:#
.lm0.ts 8,16,24,32,40,48,56,64,72
.b2.f.lm11.i-11
Subj.:##Report for $$month, $$year
.lm0
.hl 1 Accomplishments
.hl 1 Not Accomplished
.hl 1 Scheduled for Next Month
.hl 1 Problems
.hl 1 Solutions
.hl 1 Miscellaneous
.lm0.b2
[end of<MMMYY>.RNO]
```

INDEX