File

INTERNAL DOCUMENT

17

I.O.S.

ROFF TEXT FORMATTER

Version 1.0

DP/D/30

IOS Internal Document No. 17

NATURAL ENVIRONMENT RESEARCH COUNCIL

INSTITUTE OF OCEANOGRAPHIC SCIENCES

INSTITUTE OF OCEANOGRAPHIC SCIENCES

Wormley, Godalming,
Surrey, GU8 5UB.
(0428-79-2122)

(Director: Professor H. Charnock)

Bidston Observatory,
Birkenhead,
Merseyside, L43 7RA.
(051-652-2396)
(Assistant Director: Dr. D. E. Cartwright)

Crossway,
Taunton,
Somerset, TA1 2DW.
(0823-86211)
(Assistant Director: M.J. Tucker)

Marine Scientific Equipment Service
Research Vessel Base,
No. 1 Dock,
Barry,
South Glamorgan, CF6 6UZ.
(04462-77451)
(Officer-in-Charge: Dr. L.M. Skinner)

USE OF ROFF ON ACTON

The following differences exist from the usage described in the manual:

i) The command ROFF should be typed as DM3119/ROFF
The command EXPLAIN should be typed as DM3119/EXPLAIN

ii) The following file names must be used:

IN MANUAL	ON ACTON
LIBRARY/ROFF/DPMACROS	DM3119/TV/ROFF/MACROS
LIBRARY/ROFF/AMON	DM3119/TV/ROFF/AMON,

Please destroy draft versions without version number.
Reference cards to follow

```
                        **********************
                        * ROFF TEXT FORMATTER *
                        **********************

ROFF COMMAND                (see EXPLAIN ROFF COMMAND for details)

  SYSTEM ?ROFF infile [outfile] [INDEX=indexfile] [SKIP=nn] [-PAUSE]


REQUEST BREAK DEFAULT       MEANING

.af R f    no    f=1        assign format to register R, f=i,I,a,A,1,01,...
.an R +n   no    n=0        assign number to register R, R#%; if result is
                            negative, replace by zero
.ar        no    yes        arabic page numerals (syn = .af % 1)
.at R      no               assign text to register R until .en R
.bp        yes              begin page
.br        yes              break
.cc c      no    c=.        control character is c
.ce n      yes              centre next n text lines, break on each
.cn        no    yes        case normal on input
.cr        no    no         case reversed, exchange upper and lower case
                            letters on input
.ds        yes   no         double space (syn = .ls 2)
.ef n t    no    t=''''     nth even page foot title is t, 1≤n≤10
.eh n t    no    t=''''     nth even page head title is t, 1≤n≤10
.en e                       end all diversions labeled e, break if end of
                            footnote
.ep        yes              begin an even page
.fi        yes   yes        fill output lines
.fn e      no               divert text to footnotes until .en e
.fo n t    no    t=''''     nth even/odd foot titles are t, 1≤n≤10
.fs t      no    t=''''     footnote separator is t
.hc c      no               hyphenation character is c
.he n t    no    t=''''     nth even/odd head titles are t, 1≤n≤10
.hy n      no    n=3        hyphenation mode is n, 0≤n≤3
.ic c      no               insertion character is c
.ig e      no               ignore all input until .en e
.in +n     no    n=0        indent left margin n spaces
.ix e      no               divert input to index file until .en e
.ju        yes   yes        justify right margin of filled lines
.li n      no               literal, take next n lines to be text
.ll +n     no    n=60       line length is n including indent
.ls +n     yes   n=1        line spacing is n
.lv n      no               leave n consecutive blank lines; wait until next
                            page if necessary
.m1 +n     no    n=4        margin above head no. 1 is n lines
.m2 +n     no    n=2        margin below and including heads is n
.m3 +n     no    n=2        margin above and including feet is n
.m4 +n     no    n=4        margin below foot no. 1 is n lines
.mg n      no    empty      next line sets merge pattern n, 1≤n≤10
.n0        yes   yes        do not number output lines
.n1        yes   no         number output lines, reset each page
.n2        yes   no         number output lines, no page reset
.ne n      no               need room for n output lines with present
                            spacing, do .bp if necessary
.nf        yes   no         nofill, break on each input line
.nj        yes   no         no right margin justification
.np n      no    no         no printing of output for next n pages
.nu        no    yes        no first character underlining
.of n t    no    t=''''     nth odd page foot title is t, 1≤n≤10
.oh n t    no    t=''''     nth odd page head title is t, 1≤n≤10
```

REQUEST	BREAK	DEFAULT	MEANING
.op	yes		begin an odd page
.pa +n	yes	n=1	begin page with page number n
.pc c	no		parameter character is c
.pl +n	yes	n=66	paper length is n lines
.po +n	no	n=0	page offset is n, i.e. move all output n spaces right
.pr +n	no	n=0	print requests indented n, don't print if n ≤ line length
.ps +n	no	n=0	print sequence numbers of input lines indented n (n for .pr > n for .ps)
.ro	no	no	roman page numerals (syn = .af % i)
.sc c	no	c=#	shift character is c, 6-bit input only
.sk +n	no		skip at next new page to page number n
.so c/f	no		insert roff source from cat/file c/f
.sp n	yes		insert n extra spacing lines
.ss	yes	yes	single space (syn = .ls 1)
.ta	no	all	tabs set by this line or next
.tc c	no		extra tab character is c
.ti +n	yes		temporary indent, for one line only
.tr cd...	no		translate c into d on output
.uf	no	no	underline first character of each input text line
.ul n	no		underline alphanumerics in next n input text lines
.us n	no		underline all of next n input lines
.ws n	no	n=1	set window suppression mode n

The DPMACROS are available by including them with the request

```
.so LIBRARY/ROFF/DPMACROS
```

This action sets the following control characters:

```
.tr^  .ic \  .tc ~  .hc `  .pc #
```

and margins m1, m2, m3 and m4 all to 3.

```
.dp reference title sub-title author
```
set up standard document front page with the specified parameters

```
.hp heading
```
Level 1 heading on new page (indexed) - set .in 0 and .fi.

```
.h1 heading
```
Level 1 heading - can be on same page (indexed) - set .in 0 and .fi.

```
.h2 heading
```
Level 2 heading (indexed) - set .in 0 and .fi.

```
.h3 heading
```
Level 3 heading against left hand margin (indexed) - don't change .in or .fi.

```
.h4 heading
```
Level 4 heading in-line using current indentation (not indexed) - set .fi - don't change .in.

```
.ap L title
```
Set up appendix L with title

INSTITUTE OF OCEANOGRAPHIC SCIENCES

DP/D/30

DATA PROCESSING

==========

ROFF TEXT FORMATTER

Version 1.0

DP/D/30

IOS Internal Document No. 17

Tony Voss

23 August 77

CONTENTS

1. INTRODUCTION

ROFF is a text formatting program useful in the preparation of reports or publications. This manual is an example of its work. ROFF is a particularly powerful formatter and yet is easy to use in a simple way by those with only basic requirements.

Section 2 of this manual provides a basic introduction using a subset of the available facilities. This should be enough to get the majority of users started. Once you have gained confidence in this way the more advanced aspects covered in section 3 may be studied as required.

To invoke ROFF in time-sharing use the TSS command:

```
SYSTEM ?ROFF input
```

where 'input' is the file containing the source text to be formatted on to the terminal. Optional parameters may be added to specify an index file (explained later) or to cause ROFF to pause between output pages to allow non-continous paper to be inserted. As an example, to format the file A/B/C, to collect an index in file A/B/I and to pause between pages one could use the command:

```
ROFF A/B/C INDEX=A/B/I -PAUSE
```

The ROFF command has many other capabilities which are not described here as they are being extended and such a description would become rapidly out of date. To find out about the current capabilities type:

```
SYSTEM ?EXPLAIN ROFF COMMAND
```

2. FIRST STEPS

2.1 GENERAL

When processing an ordinary paragraph of text ROFF will fill the output lines with words regardless of the length of the input lines. The text can thus be prepared without consideration of the final line length. Indeed it is good practice to start each sentence on a new line, as this makes editing (particularly adding or deleting sentences) slightly easier.

INPUT: Input consists of intermixed text lines, which contain the information to be formatted, and request lines, which contain instructions about how to format it. Request lines begin with a distinguished control character. The input may be either a 9-bit (ascii) or a 6-bit (card) file.

OUTPUT: Output lines may be filled as nearly as possible with words without regard to input lineation, or may be copied one-for-one from input text. Right margin justification may be done on filled text. Computation of page numbers and line numbers is automatic; section numbers, equation numbers, etc., may also be computed. Indentation, centring, line length, line spacing, page lay-out, titling, hyphenation at line breaks, footnotes and collecting of an index are all controllable.

REQUESTS: Requests are identified by three characters at the beginning of a line. The first character must be a control character, which is normally the period or dot (.). The short table of requests below is a good starting set, adequate for routine formatting.

This entire section of the manual is reproduced in Appendix A exactly as input to ROFF. To see the effect of various commands compare this input with the formatted version, but note that some of the commands used in the preparation of this section are not explained until section 3.

There now follows a list of the subset of requests that are discussed in this chapter:

REQUEST	BREAK	DEFAULT	MEANING
.bp	yes		begin page
.br	yes		break
.ce	yes		centre next text line, do not fill it
.ds	yes	no	double space
.fi	yes	yes	fill output lines
.in n	no	n=0	indent left margin n spaces
.ju	yes	yes	justify right margins of filled lines
.ll n	no	n=60	line length is n, including indent
.nf	yes	no	nofill, break on each input line
.nj	yes	no	no right margin justification
.pl n	yes	n=66	paper length is n lines
.so c/f	no		insert ROFF source from cat/file c/f
.sp n	yes		insert n extra spacing lines
.ss	yes	yes	single space
.tr t	no		translate the character t to blank for printing
.ti n	yes		temporary indent, for next line only
.ul	no		underline alphanumerics in next input text line
.us	no		underline all of next input text line

2.2 BREAK

When a new paragraph is required it is necessary to cause a BREAK to prevent the text flowing into the next section. Breaks may be indicated specifically by the .br request or by some other command that has an implied break. The table of commands indicates whether or not a break is implied. A blank line or a line commencing with a blank also implies a break, so paragraphs are normally preserved as typed. Indented paragraphs can be produced simply by spacing the requisite amount. Some requests set values and formatting modes; values and modes in effect initially are indicated under "default".

2.3 JUSTIFICATION

Normally once the lines have been filled with as many words as they can hold extra spaces are inserted to create a straight right-hand margin, as in the previous section. For this paragraph justification has been turned off by the .nj (no justification) request.

Justification can be turned back on again by the .ju (justification) request. Note that .nj does not stop the filling of lines with words - only the subsequent padding process.

2.4 FILLING

For this paragraph the normal line filling procedure has been turned off by the .nf (no filling) request.
This means that the output lines correspond one-for-one with the input, however short or long. Indeed, it is quite possible to run right off the end of the pa
Filling is turned back on again by the .fi (fill) request.

2.5 PSEUDO-BLANKS

Sometimes it is desirable to guide ROFF in the way blanks are handled. The .tr (translate) request allows you to define character translation that is to take place before printing. This facility is explained more fully in section 3, but one use of the command is to define a 'pseudo-blank' character i.e. an additional character that is to be printed as a blank. In this manual the ^ character is normally used in this way. By using it instead of the true blank one may stop space insertion in specific places or prevent certain phrases from being split across lines. For example, if we type I O S this could turn out as I O S or even be split across two pages. By typing it as I^O^S it will always be printed as a single word I O S.

2.6 INDENTATION

Normally text is set against the left-hand margin. The use of the .in (indent) request allows one to set any other required indentation.

> This text has been indented by .in 20 to set it in 20 spaces. The indentation may be specified absolutely (as in this case) or relative to the previous value. Relative values are distinguished by the presence of an operator +-*/.
>
> This bit has had its indentation reduced by 5 with a .in -5 request, so it will be equivalent to .in 15. Now we will return to the left margin with .in 0.

Sometimes an indentation change is required for only one line. This might be achieved by separate .in requests, but often it is easier to use the .ti (temporary indent) request.

This paragraph is actually typed without any leading blanks, but the first line is indented because of a .ti 10 request preceeding it.

Now this paragraph was preceeded by .in 10 and .ti -10 requests. This results in what is called a hanging indent. This is very useful for numbered paragraphs, but unfortunately we can experience a minor nuisance if justification is in force.

(1) This is a numbered hanging indent paragraph, and ideally the second and subsequent lines will start neatly under the first word. However, justification may mess this up.

(2) This second paragraph was fixed up by starting it with (2)^^^^^^^^This... which stopped ROFF fiddling with the spacing between the) and T characters. A more advanced way of coping with this problem is by using tabulation stops, but this can wait until you wish to tackle section 3.

2.7 LINE LENGTH

The length of filled lines can be contolled by the .ll (line length) request. Here we are using .ll 20. By default a length of 60 is used, which is suitable for A4 paper and 10 characters per inch.

2.8 LINE SPACING

Spaces between lines may be introduced by simply including a blank or empty line (empty lines are ones that do not even contain a blank). Alternatively the .sp (space) request may be used to force several blank lines.

This line is separated from the previous one by .sp 4.

The .ds (double space) request has the effect of double spacing all output, as in this sample. It remains in effect until some other spacing is specified. The .ss (single space) request will return you to normal mode.

2.9 PAGE LENGTH

The .pl (page length) request allows you to set the number of lines per page. By default .pl 66 is in effect, which works for continuous 11" computer paper printed at 6 lines/inch and also for A4 paper if you are inserting each sheet in the terminal.

2.10 CENTRING

Sometimes it is required to centre text in the page. This can be achieved with the .ce (centre) request. The following is preceeded by .ce 2.

This is centred.
Filling is suspended during centring.

Text is always centred between the current indent and the right hand margin. Here we have .in 30 in effect, so .ce centres in positions 30-60:
like this!

2.11 UNDERLINES

Text may be underlined just as a typist would do it, but this is inconvenient when using most VDUs as the underline character will obliterate the text. There can also be problems when justification is applied to text underlined in this way. For example, part of this sentence was underlined as a typist might do it. These problems can be avoided by underlining each word separately. Never backspace across a space.

It is often preferable to use the .us (underscore) request to underline all of the next input line as here (noting that such continuously underscored lines are treated as a single word and cannot be

justified or split across lines) or, alternatively, use the .ul (underline) request which underlines just alphanumeric characters, omitting spaces and punctuation symbols.

2.12 NEW PAGE

ROFF will begin a new page whenever necessary. However, you may wish to do this before the previous page is full. The .bp (begin page) request achieves this.

Sometimes you find that a new paragraph or section starts too near the bottom of the page, or that a series of lines that ought to be kept together gets split across pages. This situation can be avoided by the .ne (need) request. For example

```
.ne 10
```

instructs ROFF to start a new page if less than 10 lines are available on the present page; otherwise it has no effect. By default there is an implied .ne 1 after any .sp request (or empty line) to stop a paragraph starting on the last line of the page.

2.13 SOURCE SWITCHING

The .so request causes ROFF to switch temporarily to another file for input, then return to the previous file when the other is exhausted. It is particularly useful for long documents that have been edited in pieces. For example if the pieces are in files chap1, chap2, chap3, you might prepare a very short file called book, containing only

```
.so catalog/chap1
.so catalog/chap2
.so catalog/chap3
```

then do the whole works by the simple command "roff book". This manual was produced like this, and the file containing this chapter is actually "sourced" twice - once for the main formatted occurrence and then again to provide the "as entered" copy in appendix A.

2.14 MACRO-COMMANDS

Sometimes the requests necessary to set up section headers or other standard lay-outs can become quite long and may be needed many times. A very useful feature of ROFF allows the user to define his own command sequences or macro-commands and then invoke them at will. How such macro-commands are defined will be explained in section 3, but some pre-defined macros have been provided and may be used without understanding their inner workings. This manual uses these macros extensively.

The macro definitions reside in a file LIBRARY/ROFF/DPMACROS. To use them they must first be "sourced" with the request:

```
.so LIBRARY/ROFF/DPMACROS
```

at the start of your document. This action loads the macros ready for use and also sets up certain other options, such as the use of the ^ character as a pseudo blank.

Front Page Layout Macro .dp

This macro can be used to lay out a front page like the one for this manual with only a single line of input. The request is of the form:

```
.dp reference heading sub-heading author irn
```

where each of the four arguments are separated by blanks.

reference	is the document reference number to be placed on the right hand side at the top, and on the foot of later pages.
heading	is the document name to be placed at the centre of the page.
sub-heading	is a sub-heading to be placed under the main heading.
author	is the author's name to be placed at the bottom right, above the date.
irn	is the IOS Internal Report Number, if any. If present this must be numeric, and causes the report number and an extra copy of the main reference number to be positioned correctly for the cut-out window of report covers.

A front page similar to the one on this manual could be produced by the macro-command

```
.dp DP/D/30 ROFF^TEXT^FORMATTER ^ Tony^Voss 17
```

Note that since blanks separate the arguments any blanks within an argument must be entered as a pseudo blank (^ in this case). Note also that, in the above example, no sub-heading is required, but it is necessary to provide a blank third argument so that the fourth is recognised as such. Trailing blank arguments are not needed.

By default the .dp macro sets up a standard IOS Data Processing title. Other departments may override this as required. If a MIAS document were being prepared, the lines:

```
.at (dept)
MARINE^INFORMATION^&^ADVISORY^SERVICE
.en (dept)
```

placed before the .dp request would cause that department to appear on the front page. The above sequence will not be explained further here, but is covered in section 3.

The .dp macro also arranges for the reference number, page number and current date to appear at the foot of each subsequent page.

Major Headings Macros .hl & .hp

The macros .hl and .hp may be used to set up a major heading. They take a single argument - the heading itself. The only difference between them is that .hp always starts a new page whereas .hl will start on the same page if there is plenty of room. The major heading at the top of page 2 was set up with the macro-request

```
.hp FIRST^STEPS
```

Note that the section number was assigned automatically and the heading centred, underscored and spaced above the text that follows. When using these macros there is no need to insert blank lines yourself.

Sub-Heading Macro .h2

This is similar to .hl except that it is used to set up a level 2 or sub-heading. Spacing is proportionally smaller. A sub-section number is generated automatically. The sub-heading on page 8 was set up by the macro-request

```
.h2 MACRO-COMMANDS
```

Sub-Sub-Headings Macro .h3

The .h3 macro-request generates a third level heading, without a section number. The heading immediately above this text was produced by the request

.h3 Sub-Sub-Headings^Macro^^.h3

In-line Headings Macro .h4

The lowest level of headings supported is the underscored in-line phrase:

In-line heading. This heading was generated by the macro-request:

.h4 In-line^heading.

Headings produced by .hp, .h1 and .h2 always reset the indentation to zero and turn on filling mode. A .h3 request places the heading against the left hand margin without disturbing the indentation. A .h4 heading uses the current indentation.

Appendix Macro .ap

The .ap macro-request sets up an appendix, as demonstrated by the appendices to this manual. The format is:

.ap L HEADING

where L is the appendix letter (usually A B C etc., but could be anything) and HEADING the heading. Note that appendices have their pages numbered independently from the main part of the document.

The .ap macro-request is used in place of the .hp or .h1 requests, but sub-headings within an appendix may be generated with .h2, .h3 or .h4.

2.15 INDEX

It is possible to arrange for the automatic generation of an index or list of contents. How you can do this is explained in section 3. However, if you are using the macros described in section 2.14 index entries are made automatically. The .hp, .h1, .h2, .h3 and .ap headings will be indexed, provided you have specified an index file when you invoked ROFF. The index file so produced contains ROFF requests and must itself be ROFFed to get the final product. The list of contents for this manual was produced in exactly this way.

3. ADVANCED TOPICS

This section covers the more advanced aspects of ROFF. The topics are arranged in an order suitable for digestion, gradually getting more demanding on the reader. Some sections assume familiarity with the contents of previous ones.

3.1 PAGE NUMBER CONSIDERATIONS

If a document is to eventually be printed on both sides of the paper it may be necessary to distiguish between even and odd pages. For example, it is usual to start a new chapter on the right hand page of a book (odd number).

The .op request can be used to skip to the start of the next odd page, while the .ep request can be used to ensure an even page number.

The .pa request can be used to start a new page with a specified number. Thus .pa 20 would start a new page and number it 20, while .pa +20 would start a new page and number it 20 more than the current one.

To set the number of the next page without forcing a new page the .sk (skip) request can be used. .sk +1 would cause a page number to be skipped when the next new page occurs.

Page requests are only obeyed if they are necessary. Thus a .bp request will be obeyed only if there is something on the current page. To actually get a blank (but titled) page (perhaps for a diagram insertion) it would be necessary to have:

```
.bp
.sp
.bp
```

If the DPMACROS are being used the request

```
.an (%%) 2
```

placed before "sourcing" the DPMACROS will cause them to operate in two page mode.

3.2 TOP/BOTTOM MARGINS

The requests .m1, .m2, .m3 and .m4 control the margin space to be left at the top and bottom of each page.

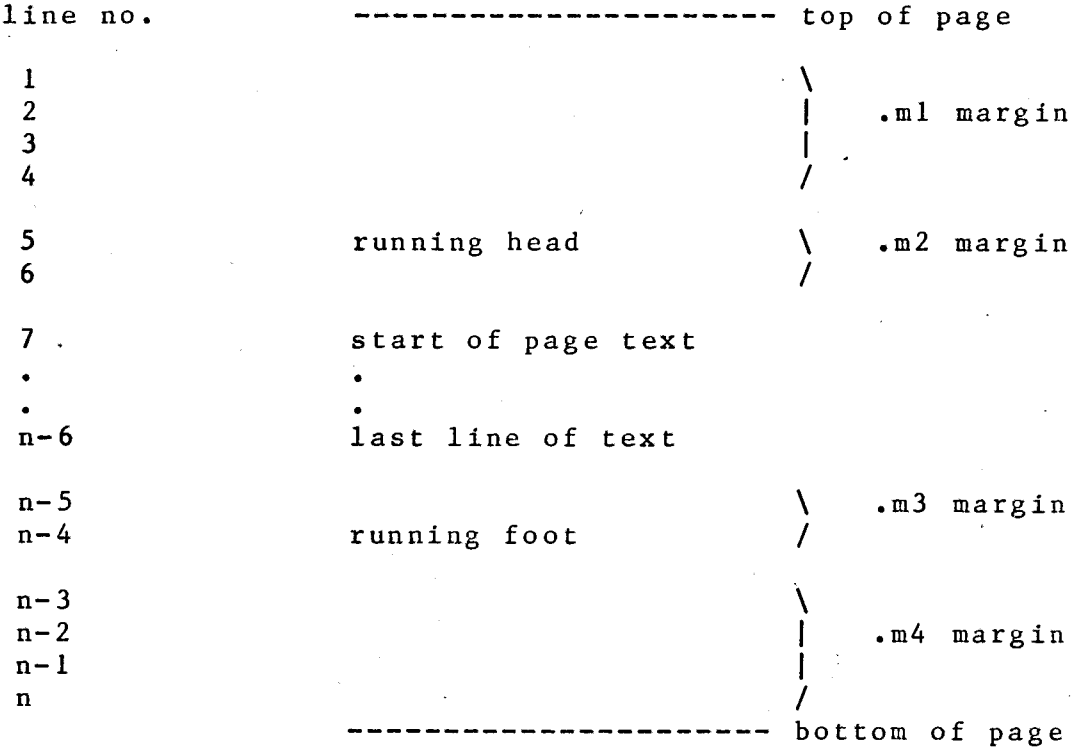
The .m1 request specifies the number of lines to be left at the top of the page above any page heading (running head). The default value is 4.

The .m2 request specifies an additional top margin in which may be placed a running head (described below). The default value is 2, which is suitable for a single line running head.

The .m3 request specifies an upper bottom margin wich includes any running foot. The default value is 2.

The .m4 request specifies a lower bottom margin to occur below the .m3 margins. The default is 4.

The relationship of these various margins can be visualised thus:



The actual number of lines available for use on the page is the value of the paper length (.pl) less the sum of the .m1 - .m4 margins.

3.3 HEADINGS AND FOOTINGS

Running heads and foots may be set up by the .he and .fo requests. The format of theses requests is:

.he n 'part1'part2'part3'

Where n is a number in the range 1 to 10 or is omitted (which implies n=1). Part1 of the heading is left justified, part2 centred and part3 right justified with respect to the margins current when the title was set. Any non-blank character may be used instead of the ' delimiter. In headings and footings only, any occurrence of the % character in a title will be replaced by the current page number when the title is printed.

Up to 10 separate lines of heading or 10 lines of footing can be defined, the n parameter being used to specify the n'th line. Headings appear in the space left by the .m2 margin in the order 1,2,...10. Footings appear in the space left by the .m3 margin in the order 10,9,...1. If the margin is of size m where m<10, titles m+1 to 10 do not appear.

The .eh and .oh requests may be used instead of .he to set up headings for even and odd pages respectively, and the .ef and .of requests instead of .fo for even and odd footings. The main use of this facility is to provide "mirror image" titles for odd and even pages; eg.

```
.of 'DP/D/30'Page %'IOS'
.ef 'IOS'Page %'DP/D/30'
```

3.4 KEEPS AND FIGURES

The use of the .ne request to keep a number of lines on the same page has already been described in section 2.12. Another use of .ne would be to leave some blank space for the insertion of a diagram. If 30 lines were needed for this the sequence

```
.ne 30
.sp 30
```

would achieve this. However, up to 29 lines might also be left on the preceding page. This situation can be avoided by the .lv (leave) request.

```
.lv 30
```

would leave 30 consecutive blank lines as before but bring text forward to fill up any remaining space on the current page if necessary.

3.5 WINDOWS

Automatic formatting programs often start a new paragraph right at the bottom of the page (leading window) or allow the last line of a paragraph to appear at the top of a new page all on its own (trailing window). ROFF is able to ease this problem by not starting after .sp (or a blank line) if within a pre-defined number of lines of the bottom. To avoid trailing windows ROFF can temporarily reduce the bottom margin (defined by .m4) by one in order to squeeze on the last line of a paragraph. Window suppression may be controlled by the .ws request:

```
.ws 0  Do not suppress windows
.ws 1  Suppress 1-line leading and trailing windows (default)
.ws 2  Suppress 2-line leading and 1-line trailing windows
.ws 3  Suppress 3-line leading and 1-line trailing windows
```

There is no check for paragraphs that are so small that they could fit on the bottom of a page even though they start very near the bottom. They will be forced on to the next page. The action can be best understood if .sp is regarded as being equivalent to the following sequence (assuming single line spacing and .ws n in effect):

```
.m4 -1
.sp
.m4 +1
.ne n
```

The bottom margin is reduced by one, making space for one extra line of text. The .sp action is then performed, which flushes out the last line of the paragraph on to the page. The bottom margin is then restored and the .ne forces a new page if within n lines of the bottom.

3.6 TABULATION

Tab characters may be used to align output on specified character positions. Alignment is immune to padding during justification.

Tabulation may be indicated by the ASCII TAB character itself (generated with the TAB key if your terminal has one; otherwise CNTL I) or by a pseudo tab character set by the .tc request. By default no pseudo tab character is set, but if you are using the DPMACROS

```
.tc ~
```

is in effect. In this manual ~ will be used for this purpose.

When preparing text on the HP2640 terminals for block mode entry any TAB key actions are immediately expanded to blanks in the terminal. If it is desired to use the ROFF tab features a pseudo tab must be used instead.

Tab positions or stops are specified by the .ta request. They may function just like a traditional typewriter stop (which left aligns the following text) or may be used to centre a field or right align a field. The .ta request may be made in one of several forms.

In one form the .ta request is qualified by the character positions (relative to the current indentation) in which the stop is to be set. Each position may further be qualified by the letter L C or R, which indicates the field is to be left, centre or right aligned. In the following example we wish to lay out some equations so that the left hand sides are right aligned, the equals signs are left aligned and the equation numbers are right aligned. Notice that a TAB character is required wherever spaces are to be inserted, which includes the space at the left hand margin before the first aligned field.

```
.nf
.tc ~
.ta 15R 17L 60R
~sin x~= x - x**3/3! + x**5/5! - ...~(9)
~sinh x~= x + x**3/3! + x**5/5! + ...~(10)
```

The result is:

```
        sin x = x - x**3/3! + x**5/5! - ...          (9)
       sinh x = x + x**3/3! + x**5/5! + ...          (10)
```

Tabs may also be set relative to the previous position. For example, to divide the page into 8-character columns one could use:

```
.ta 1 +8 +8 +8 +8 +8 +8 +8 +8
```

This next one sets left- and right-justified stops at 60 and 54:

```
.ta 60L -6R
```

If you prefer to think visually rather than in terms of character positions the numbers may be omitted from the .ta request altogther, in which case the line following is expected to contain just the letters L C or R where the appropriate stops are to be set. The table of requests on page 3 was set up using:

```
.ta
L          R     .C    L
```

On page 5 it was mentioned that hanging indents could be controlled using tab stops. The paragraph numbered (2) could have been written:

```
.ta 11
.in 10
.ti -10
(2)~This second paragraph....
```

Notice that the .in request specifies how many positions to omit, but the .ta request specifies where the next character is to be placed. The tab stop must thus be one greater than the indentation.

3.7 HYPHENATION

In order to fit words onto the line when in fill mode ROFF may hyphenate words. The boldness with which this is done can be controlled by the .hy request.

.hy 0 — Specifies that no words are to be split.

.hy 1 — Specifies that words may be split only at an existing hyphen sign or at a hyphenation character (explained below)

.hy 2 — Words may be split as with .hy 1 and also before certain suffixes.

.hy 3 — Words may be split as with .hy 2 and also between certain pairs of letters. This is the default setting.

It should be noted that the shorter the line length the more necessary it becomes to allow hyphenation, as newspaper readers are well aware.

Inevitably mistakes occur from time to time. Watch the effect of reducing the line width to 6 letters on the following:

```
cesspool
courthouse
tetrabromomethane

.ll 6

ces-
spool
cour-
thouse
tetrabromomethane
```

In the first two words ROFF picks on letter pairs that, unfortunately, are not on syllable boundaries. In the last one ROFF has insufficient knowledge of chemistry to split it at all.

For such cases we may use the .hc request to specify a hyphenation character with which we may indicate where ROFF may hyphenate words. If you are using the DPMACROS the ` character is set up for this purpose.

```
.hc `
```

Now we can supply syllibification for the unfortunate tetra`bromo`meth`ane. Watch what happens with line lengths of 18, 12 and 6 characters:

```
.ll 18

tetrabromomethane

.ll 12

tetrabromo-
methane

.ll 6

tetra-
bromo-
meth-
ane
```

Words containing the hyphen sign or the hyphenation character are never split in other places. This convention permits a trick for completely avoiding hyphenation of a selected word: put a hyphenation character at the beginning or end of the word. This is useful for fixing up words that get split across pages or for proper names that do not look right hyphenated: Constantinople` will never be split.

3.8 TRANSLATION

On page 5 the use of the .tr request to translate a character into a blank was described. Actually the .tr request is rather more powerful than this. Its full form is:

```
.tr aibjckdl...
```

which specifies that 'a' is to be translated to 'i', 'b' to 'j' and 'c' to 'k' etc. If an odd number of letters is specified a blank is assumed to complete the last pair. Hence our earlier form:

```
.tr ^
```

to specify a pseudo blank. All characters initially translate into themselves and a translation not mentioned in the .tr request is unaffected. To turn off translation it is necessary to request translation to itself:

```
.tr ^^
```

For an example of these aspects study appendix A to see how the discussion on page 5 was produced, bearing in mind that ^ is normally translated into a blank.

3.9 MERGING

After an output line has been prepared it may be merged with any fixed information required, known as a merge pattern. Up to 10 merge patterns may be set by the .mg request. The line following the request is taken to be the desired merge pattern. In order to perform the merging ROFF replaces every true blank of every non-empty output line with the corresponding character of merge pattern 1. Any remaining blanks are then replaced from merge pattern 2 and so on. To demonstrate this we will set up a table which describes some of the uses of merge patterns.

USE	DESCRIPTION
TABLES	This is itself a demonstration of the use of merge patterns in a table.
INDEX	The . . . . pattern in the index of this manual was produced by a merge pattern.
UPDATES	Vertical bars might be used in a margin to indicate text that has been updated.

The vertical lines were inserted with a merge pattern. The horizontal lines were entered in the text. Note that the first and last lines were continuous, while the intermediate horizontal lines had spaces left in them to allow the vertical lines to cut through. Tabs were used to centre information in the first table column and left justify the entry in the second column.

This text is demonstrating the update bar use. It was set up by:

```
.ta 65
.mg
~|
```

3.10 LITERALS

Occasionally it might be required to enter lines beginning with the ROFF request character. This can be done by specifying that the line is to be taken literally by using the .li (literal) request.

```
.li 2
.sp This and the next line which starts with a
. could not be printed without .li.
```

Actually the character used to introduce a ROFF request (normally .) can be set to anything with the .cc (control character) request.

In this manual the examples of requests that are to be printed rather than obeyed have been entered using another character that is translated into .; this is another way to avoid the problem.

3.11 PAGE OFFSET

The .po (page offset) request can be used to shift the entire page if it is printed too close to the left hand margin. We have just issued .po +5.

3.12 TROUBLE-SHOOTING

ROFF does not issue error messages (apart from file access or command line errors) but tries to do a reasonable thing whatever the request. A number of facilities are available to assist in tracing problems.

Input line numbers can be printed using the .ps (print sequence) request. The request specifies the column number in which the numbers are to be placed.

.ps 62

Now we are printing the line numbers in position 62, which is just in the right hand margin.

Output line numbers can be printed by the .nl or .n2 requests. The numbers are placed in the left hand margin reserved by the .po request, so this must be sufficient (usually .po 4 will suffice). The request .nl causes lines on each page of output to start again from 1, while .n2 causes the numbering to continue from the request onward. Output line numbering is turned off by the .n0 request.

Original line starting positions can be determined by using the .uf (underline first) request that causes the first character of each input line to be underlined in the output, as here. This option is turned off by the .nu (no underlining) request.

Requests can be printed as they are obeyed by the .pr (print request) request. The .pr request specifies the position where the request is to be printed. This must be in the right hand margin and must be greater than the position specified for the .ps request. Here we have .pr 65 in effect.
.b
Notice that when we encounter a break, as here, it is obeyed and printed in the margin. Often several requests have to be printed against one line of output. Invoking a macro-command will cause an attempt to print all the implied commands. Now we are going back to normal by doing .pr 0, .ps 0, .nu and .po -5.
.p

3.13 DIVERSIONS

Certain requests, namely .at, .fn, .ig and .ix, cause the output to be diverted elsewhere. Each diversion should have associated with it a label, which may be blank or have the form of a register name (see below). The diversion continues until a corresponding .en (end) request. Nested diversions require non-empty labels. Various forms of diversion are used in the following sections.

3.14 FOOTNOTES

Lines diverted by a .fn (footnote) request are formatted normally but are held for the bottom of the page. Line formatting is handled quite independently in the body and footnote text. In particular, separate settings are maintained for .fi, .ce, .in, .ju, .ll, .ls, .mg, .ta, .ti, their synonyms and antonyms. In order to set these parameters for footnotes it is necessary to divert there the required requests.

Footnotes are separated from the body of the text by a footnote separator. This is set up using the .fs (footnote separator) request.* Although ROFF attempts to place the footnote on the same page on which it was defined it can happen that some or all of the note gets pushed over on to the next page. Similarly the line containing the reference to the footnote (* in our example) could fall on a different page. (Note that there is no connection between the footnote and the readers' reference to it as far as ROFF is concerned.) These problems can all be avoided by using a .ne request to ensure sufficient space. The present example was achieved with:

```
.fs '----------------------'''
Footnotes are separated from the body of the text by a
footnote separator.
.ne 9
This is set up using the .fs (footnote separator) request.*
.fn
.ll 50
* The format of a .fs request is exactly.....
                        ...side the footnote.
.en
```

---

* The format of a .fs request is the same as the page heading or footing requests. In this example we also diverted a .ll 50 request to achieve a shorter line length inside the footnote.

3.15 REGISTERS AND INSERTIONS

Numerical values or text fragments can be stored in registers. Each register has a 1-to-4 character name chosen from a-z, 0-9, % and #, which must be enclosed in parentheses unless it is a single character name (when enclosure is optional). Upper and lower case are identical in register names. Examples of register names would be:

VALID	INVALID
a	&
(A#)	a#
(AbCd)	(IJKLM)
(%)	AB

In every input line appearances of an insertion character followed by a register name are replaced by the contents of that register. The insertion character is set by the .ic request. If you are using the DPMACROS the insertion character is set to \, otherwise it is initially undefined.

An insertion character not followed by a register name disappears and the following character is left untouched - even when that is the insertion character also. Insertion lines are effective everywhere else, including request lines, diversions and the interior of parenthesised register names.

It may be instructive to puzzle out the reason for the behaviour of the following sequence:

```
.ic \          make \ the insertion character
.ic \          turn off the insertion character
.ic \          make \ the insertion character
.ic \\         make \ the insertion character
```

3.16 NUMBER REGISTERS

The request .an R n assigns the number n to register R. The number must be a decimal integer, possibly preceded by one of +-*/ to cause modification based on the old value. Numeric values are not allowed to go negative; negative values are replaced by zero. Each register containing a number has a format for printing in insertions. The request .af R f assigns format f to register R. Possible formats and numbers printed in each case are:

```
1    0,1,2,3,4,...,26,27,28,...
01   00,01,02,03,04,...,26,27,28,...
i    0,i,ii,iii,iv,...,xxvi,xxvii,xxviii,...
I    0,I,II,III,IV,...,XXVI,XXVII,XXVIII,...
a    0,a,b,c,d,...,z,aa,bb,...
A    0,A,B,C,D,...,Z,AA,BB,...
```

Formats 1,01,001,... show a minimum width for decimal printing. Shorter numbers in these formats will be padded with leading zeroes.

The registers listed below contain special values; all others start at 0. All registers have format 1 initially.

(year), (mon), (day)	- date and time of day of the start
(hour), (min), (sec)	of the present run, coded as numbers
(%)	- current page number
(#)	- current line number

Right after .bp, .ep, .op, or .pa, register % contains the current page number. Otherwise it contains the number of the page to which the last character of the last preceding line of input text would be assigned, except in pathological cases where certain requests (.m1, .m2, .m3, .m4, .ne, .pl, .sk) unpredictably alter pagination.

Note: the % in titles is not preceded by an insertion character, and is evaluated on output rather than upon input.

Register # contains the line number of the last character of the last preceding line of input text. The request .an # n (after a break) will cause the next line to be numbered n+1.

3.17 TEXT REGISTERS

The .at (assign text) request diverts text into the specified register until a corresponding .en request is encountered. The diverted text cannot exceed 400 characters. The final carriage return in the diverted text is dropped. Text registers can be useful if a fragment of text is used repeatedly in many places. For example we could set up a register like this:

```
.at (IOS)
Institute of Oceanographic Sciences
.en (IOS)
```

Assuming an insertion character of \ is in effect, such phases as

"The \(IOS) has four main sites..."

will be turned in to

"The Institute of Oceanographic Sciences has four main sites...".

If you are using the DPMACROS the following registers will have been set up for you:

(date) Contains the current date in the form used at the foot of this page. This footing is set up when the .dp request is made, so if you want some other date there you can redefine (date) before invoking the .dp macro.

(dept) This contains the department and is used by the .dp macro to set up the front page. It should now be apparent how the procedure given on page 9 works.

(titl) This contains the major title supplied to the .dp macro request (for this manual it is "ROFF TEXT FORMATTER").

(tit2) This contains the sub-title supplied to the .dp macro request.

3.18 CROSS-REFERENCES

You may have noticed that this manual sometimes refers back to a previous page or section and gets it right. This is done by means of a cross-reference defined on the page to which we wish to refer.

For example back in the input text for page 7 there is the number assignment:

```
.an (XR9) \%
```

You will recall that register % contains the current page number, so the above request assigns it to (XR9). We can now refer to that page as \(XR9) and get the right number.

If you are using the DPMACROS the current .h1 (or .hp), .h2, and .h3 section numbers are available in the registers a, b, and c.

If it is required to refer to a subsection it may be preferable to assign the full reference to a text register to save having to assign section and sub-section numbers to separate number registers. Back on page 7 we also had in the input text:

```
.at (XR4)
\a.\b
.en (XR4)
```

Now "section \(XR4)" expands to "section 2.12".

Backward references are thus fairly easy, but there are very real obstacles with forward ones. One possibility is to ROFF the entire document twice by the following sequence:

```
.pa 1
.np 9999
.so user/document
.np 0
.pa 1
.so user/document
```

This sequence uses the .np (no print) request to suppress output printing for the first pass while collecting forward references. Printing is then turned back on again, the page number reset and the document formatted a second time. This is all very well but small changes in pagination could occur between the two passes depending on the length of the numbers inserted. Also it is almost twice as costly in computer time.

3.19 INDEX ENTRIES

The diversion request .ix (index) causes all input up to a matching .en rquest to be diverted to the file specified for index entries when ROFF was invoked. If no file was specified this text is lost. The diverted text is not changed in any way except that insertions are obeyed.

Suppose we wish to index the phrase "keyword" by page number. When the required phrase occurs we can include the diversion:

```
.ix
"keyword" appears on page \%
.en
```

As the text is diverted the current page number is inserted as requested.

The index can be built up exactly as it is to appear, or can include ROFF requests so that it can itself be ROFFed. If an alphabetical index is required this can be produced by sorting the resulting file - perhaps by using the sort command in the QED editor.

If you are using the DPMACROS the .hp, .h1, .h2 and .ap requests all make index entries for you. The layout of the index file is set up by diverting appropriate ROFF requests when the .so LIBRARY/ROFF/DPMACROS request is obeyed.

3.20 IGNORE

The diversion .ig (ignore) causes all input up to the matching .en to be lost. One obvious use is to allow insertion of comments:

```
.ig
This document should be ROFFed, not listed
.en
```

If the above sequence were placed at the start of a ROFF file it would be seen by anyone listing the file but would not appear in the ROFFed version.

The .ig request can also be used to include text conditionally. The following sequence turns into "January" or "February" depending on whether the current month is number 1 or 2:

```
.ic \
.ig (\(mon))
.en (1)
January
.ig (0)
.en (2)
February
.en (0)
```

ROFF carries the month number in (mon). If (mon) is 2, ROFF sees .ig (2) and ignores everything down to .en (2), so "February" gets taken as input and .en (0), which ends nothing, is ignored. On the other hand, if (mon) is 1 ROFF ignores only down to .en (1) and "January" gets included. The .ig (0) on the next line causes "February" to get bypassed.

A complete sequence which assigns the current month to the register (amon) is available in the file LIBRARY/ROFF/AMON - this is actually used by the DPMACROS when setting up the register (date) as used at the bottom of this page.

Yet another use of the .ig request might be to conditionally include or exclude text depending in the version number of the document to be produced. It should be possible to define a macro (see later) to declare the version level of a given bit of input.

3.21 MACRO-COMMANDS

When the contents of a text register are inserted in the text any ROFF requests in that register are obeyed. We might set up the following sequence for starting a new paragraph:

```
.at (ph)
.sp
.ne 3
.ti 10
.en (ph)
```

Whenever we want to start a new paragraph we can now do so with:

```
\(ph)
```

This feature is so useful that, in the case of two letter text registers only, ROFF allows us to invoke them like a request:

```
.ph
```

Now we are able to define our own macro-requests.

A request may take a number of arguments on the same line. Each argument is separated from the previous one by at least one blank. A .sp 10 request has, for example, an argument of 10. We can arrange for arguments to our macro-requests to be inserted in the expansion by the presence of a parameter character followed by the argument number. The parameter character is defined by the .pc (parameter character) request. If you are using the DPMACROS this is set to #. Thus #1 is replaced by the first argument, #2 by the second and so on. The parameter character only behaves as such inside a macro - not in ordinary text.

Now we could enhance our .ph macro to allow user selection of the indentation:

```
.at (ph)
.sp
.ne 3
.ti #1
.en (ph)
```

This version can be invoked by requests like .ph 10 or even .ph -10.

You should now be in a position to design your own macro-requests. The DPMACROS are available as a starting point.

If you want to modify one but use the others you can redefine just that one - ROFF uses the most recent definition of a request. We will now look at the .h2 macro and see how it works. The macro is listed here with line numbers for easy reference and comments which do not appear on the real thing. Note the use of " to introduce a comment.

```
.at (h2)                        "1   start definition of h2
.sp 2                           "2   space down to clear last text
.ne 10                          "3   new page if near bottom
.an b +1                        "4
.an c 0                         "5
.in 0                           "6   reset indentation
.us                             "7   underscore heading line
\\a.\\b^^#1                     "8   output heading
.sp 2                           "9   space again
.fi                             "10  turn on fill mode
.ix h                           "11  start diversion to index
^^^^\\a.\\b^^#1^^^^\\%          "12  index entry
.en h                           "13  end of index diversion
.en (h2)                        "14  end of h2 definition
```

.h2 is a level 2 heading, so line 4 bumps the sub-section count held in number register b. Line 5 sets the sub-sub-section count to zero, so that the next use of .h3 will bump c to 1. Line 8 outputs the heading itself, complete with section numbers and the user's title. Note the use of double insertion characters here: this is necessary to delay the insertion until macro-expansion time. Insertion characters are obeyed whenever encountered and if only a single one had been present the insertion would have been made when the macro was first encountered. Lines 11 to 13 define the index entry. This is spaced out to the appropriate position. The diverted text has any insertions made, but is not otherwise formatted at this stage. The tab character thus goes into the index file unchanged and its subsequent expansion is according to the format defined for the index file. The current page number is also inserted in this entry. The index entry is merged with a dot pattern. Note the use of pseudo-blanks to stop dots being placed right against the title and page number.

4. ODDS AND SODS

4.1 CARD INPUT

In case there are any hardy individuals left who want to input text on cards, ROFF will accept such input and provides some assistance for getting a reasonable result.

Each character is normally treated as lower case unless it is immediately preceded by a shift character. Normally the shift character is #, but it may be changed by the .sc request.

Card code	0	1	2	3	4	5	6	7	8	9	[	#	@	:	>	?
Lower case	0	1	2	3	4	5	6	7	8	9	[	#	@	:	>	?
Upper case	0	1	2	3	4	5	6	7	8	9	[	#	@	:	>	?
Card code	sp	A	B	C	D	E	F	G	H	I	&	.	]	(	<	\
Lower case	sp	a	b	c	d	e	f	g	h	i	&	.	]	(	<	\
Upper case	ht	A	B	C	D	E	F	G	H	I	&	.	]	.	<	~
Card code		J	K	L	M	N	O	P	Q	R	-	$	*	)	;	
Lower case	\	j	k	l	m	n	o	p	q	r	-	$	*	)	;	´
Upper case	ff	J	K	L	M	N	O	P	Q	R	bs	$	@	}	;	`
Card code	+	/	S	T	U	V	W	X	Y	Z		,	%	=	"	\
Lower case	+	/	s	t	u	v	w	x	y	z	_	,	%	=	"	\
Upper case	nl	/	S	T	U	V	W	X	Y	Z	_	,	%	=	"	\

bs = backspace, ff = formfeed (not useful), ht = horizontal tab, nl = newline, sp = space

4.2 CASE REVERSAL

If you have to work from a terminal that has no lower-case characters you could consider preparing a BCD input file with the card input conventions. yOU MIGHT ALSO FIND THE .CR (CASE REVERSED) REQUEST USEFUL. tHIS CAUSES THE CASE OF ALL ALPHABETIC CHARACTERS TO BE SWITCHED ON INPUT, AS HERE. This sentence was typed in upper case except for the very first letter, which was lower case. cASE REVERSAL IS SWITCHED OFF BY THE .CN (CASE NORMAL) REQUEST.

4.3 PROCESSING LOGIC

Steps 1 to 6 are performed character-by-character.

1. Read input from current source (file, text register, number register, or insertion argument). If exhausted, pop source and try again.
2. If source is cards, replace shift-character sequences. If source is text register, recognize parameter flag (see step 7) and switch source to argument.
3. Reverse case under .cr.
4. Recognize insertion character, continue getting characters for name, then switch source (further insertions are honored within a parenthesized name).
5. Replace tab character by unpaddable blanks according to .ta, except under .ix or .at.
6. Repeat steps 1-5 until one line of input is collected.

Steps 7 to 10 are performed input-line-by-input-line.

7. Inside .at, .ig, and .ix diversions, recognise closing .en; otherwise diverted line goes straight to destination. Under .at, parameter positions as indicated by .pc are replaced by parameter flags.
8. Recognize and perform requests.
9. Underline according to .ul, .us and .uf and replace hyphen characters by hyphenation flags.
10. Repeat steps 1-9 until a line of output is collected.

Steps 11 to 15 are performed output-line-by-output-line.

11. Eject a page if there is not room for the line and insert footnotes, margins, and titles as appropriate.
12. Insert line numbers and request summaries as requested by .n1, .n2, .ps, and .pr.
13. Insert combined merge patterns 1,2,3,...,10.
14. Translate according to .tr.
15. Append the line, preceded by spacing specified by .ls, onto the output file, or onto the footnote collection buffer if under .fn, or nowhere if under .np.

4.4 CAVEATS

Some boundary conditions and pitfalls:

- ROFF imposes a limit of 180 characters, counting backspaces, upon each line (or pair of lines in filled text), limits text registers to 400 characters each, and limits the footnotes on any one page to 4000 characters.
- Although it accepts all ascii characters and escape sequences, ROFF knows the meaning of only a few - carriage return, new line, horizontal tab and backspace. In particular, forward and reverse half line feeds will work reasonably only if balanced within the line.

- Control characters usually don't work in merge patterns.
- Backspacing across a blank in filled text almost never produces the desired effect.
- No numeric value is ever permitted to go negative. In particular a negative cumulative indent or a negative register value can not exist.
- .ul and .ce apply only to text lines, not to titles.
- Titles, registers, footnotes, tab settings, and merge patterns use memory and cause ROFF to grow in space and cost.

4.5 ACKNOWLEDGEMENTS

ROFF was obtained by IOS (under Western Electric licence no. IOS-011477) from:

Bell Telephone Laboratories,
600 Mountain Avenue,
Murrey Hill,
N.J. 07974
U.S.A.

The IOS version is functionally unchanged in respect of the requests and the formatting, but has been updated to make the ROFF command and files conform to IOS prefered practice.

Parts of this manual were derived from BTL's manual MHCC - 005.

APPENDIX A

SAMPLE INPUT TO ROFF

This appendix contains the entire input that was required to produce section 2 of this manual. It forms a useful comparison with the formatted version. The frequent use of the .an request is assigning cross-references for use in appendix B. To produce a true copy of what was entered, all translations have been turned off, so the pseudo-blanks in the headings and footings are visible.

```
.tr |^{.
.hp FIRST^STEPS
.an (XR1) \%
.h2 GENERAL
When processing an ordinary paragraph of text ROFF will fill the output
lines with words regardless of the length of the input lines.
The text can thus be prepared without consideration of the final line le
Indeed it is good practice to start each sentence on a new line,
as this makes editing (particularly adding or deleting sentences)
slightly easier.
.h4 INPUT:
Input consists of intermixed
.ul
text lines,
which contain the information to be formatted,
and
.ul
request lines,
which contain instructions about how to format it.
Request lines begin with a distinguished
.ul
control char`ac`ter.
The input may be either a 9-bit (ascii) or a 6-bit (card) file.
.h4 OUTPUT:
Output lines may be
.ul
filled
as nearly as possible with words without regard to input
lineation, or may be copied one-for-one from input text.
Right margin justification may be done on filled text.
Computation of page numbers and line numbers is automatic;
section numbers, equation numbers, etc., may also be computed.
Indentation, centring, line length, line spacing, page
lay-out, titling, hyphenation at line breaks,
footnotes and collecting of an index
are all controllable.
.sp
```

```
.h4 REQUESTS:
Requests are identified by three char`ac`ters at the beginning of a line
The first char`ac`ter must be a control char`ac`ter, which
is normally the period or dot (.).
The short table of requests below is a good starting set, adequate
for routine formatting.
.sp
This entire section of the manual is reproduced in Appendix^A exactly
as input to ROFF.
To see the effect of various commands compare this input with the
formatted version,
but note that some of the commands used in the preparation of this secti
not explained until section^3.

There now follows a list of the subset of requests that are discussed
in this chapter:
.sp
.hy 0
.ne 22
.ul
REQUEST BREAK DEFAULT        MEANING
.an (XR6) \%
.sp
.ta
1          r     c    1
.nj
.in+22
.cc/
/ti-22
.bp~yes~~begin page
/ti-22
.br~yes~~break
/ti-22
.ce~yes~~centre next text line, do not fill it
/ti-22
.ds~yes~no~double space
/ti-22
.fi~yes~yes~fill output lines
/ti-22
/ti-22
.in n~no~n=0~indent left margin n spaces
/ti-22
.ju~yes~yes~justify right margins of filled lines
/ti-22
.ll n~no~n=60~line length is n, including indent
/ti-22
.nf~yes~no~nofill, break on each input line
/ti-22
.nj~yes~no~no right margin justification
/ti-22
.pl n~yes~n=66~paper length is n lines
/ti-22
.so c/f~no~~insert ROFF source from cat/file c/f
/ti-22
```

```
.sp n~yes~~insert n extra spacing lines
/ti-22
.ss~yes~yes~single space
/ti-22
.tr t~no~~translate the character t to blank for printing
/ti-22
.ti n~yes~~temporary indent, for next line only
/ti-22
.ul~no~~underline alphanumerics in next input text line
/ti-22
.us~no~~underline all of next input text line
/cc.
.hy 3
.in 0
.ju
.h2 BREAK
.an (XBR) \%
When a new paragraph is required it is necessary to cause a BREAK to
prevent the text flowing into the next section.
Breaks may be indicated specifically by the .br request
or by some other command that has an implied break. The table of command
indicates whether or not a break is implied.
A blank line or a line commencing with a blank also implies a break, so
paragraphs are normally preserved as typed.
Indented paragraphs can be produced simply by spacing the requisite amou
Some requests set values and formatting modes; values
and modes in effect initially are indicated under "default".
.h2 JUSTIFICATION
.an (XJU) \%
.nj
Normally once the lines have been filled with as many words as they can
hold extra spaces are inserted to create a straight right-hand margin,
as in the previous section. For this paragraph justification has been
turned off by the .nj (no justification) request.
.sp
Justification can be turned back on again by the .ju (justification) req
Note that .nj does not stop the filling of lines with words - only
the subsequent padding process.
.ju
.h2 FILLING
.an (XFI) \%
.nf
For this paragraph the normal line filling procedure has been
turned off by the .nf (no filling) request.
This means that the output lines correspond one-for-one
with the input, however short
or long. Indeed, it is quite possible to run right off the end of the pa
Filling is turned back on again by the .fi (fill) request.
.h2 PSEUDO-BLANKS
.an (XR7) \%
Sometimes it is desirable to guide ROFF in the way blanks are handled.
The .tr (translate) request allows you to define character translation
that is to take place before printing. This facility is explained
more fully in section^3, but one use of the command is to define a 'pseu
```

```
character i.e. an additional character that is to be printed as a blank
.tr !^
In this manual the ! character is normally used in this way.
By using it instead of the true blank one may stop space insertion
in specific places or prevent certain phrases from being split across
lines. For example, if we type I^0^S this could turn out as I^^^0^^S
or even be split across two pages. By typing it as I!0!S it will
always be printed as a single word I^0^S.
.tr !!
.h2 INDENTATION
.an (XIN) \%
Normally text is set against the left-hand margin.
The use of the .in (indent) request allows one to set any other requirec
indentation.
.br
.in 20
.ne 10
This text has been indented by .in^20 to set it in 20 spaces.
The indentation may be specified absolutely (as in this case)
or relative to the previous value.
Relative values are distinguished by the presence of an operator +-*/.
.br
.in -5
This bit has had its indentation reduced by 5 with a .in^-5 request,
so it will be equivalent to .in 15.
Now we will return to the left margin with .in 0.

.in 0
Sometimes an indentation change is required for only one line.
This might be achieved by separate .in requests, but often it is
easier to use the .ti (temporary indent) request.
.sp
.ti 10
This paragraph is actually typed without any leading blanks,
but the first line is indented because of a .ti^10 request preceeding it

.in 10
.ti -10
Now this paragraph was preceeded by .in^10 and .ti^-10 requests.
This results in what is called a hanging indent.
This is very useful for numbered paragraphs, but unfortunately we can ex
a minor nuisance if justification is in force.

.ti -10
(1)      ^^This is a numbered hanging indent paragraph, and ideally
the second and subsequent lines will start neatly under the first
word. However, justification may mess this up.

.tr !^
.ti -10
(2)^^^^^^^This second paragraph was fixed up by starting it with
.an (XR7) \%
(2)!!!!!!!This... which stopped ROFF fiddling with the spacing
between the )^and^T characters.
```

```
.tr !!
A more advanced way of coping with this problem is by using tabulation
.an (XR7) \%
stops, but this can wait until you wish to tackle section^3.
.ne 16
.h2 LINE^LENGTH
.an (XLL) \%
.ll 20
The length of filled lines can be contolled by the .ll (line length)
request. Here we are using .ll^20.
By default a length of 60 is used, which is suitable for A4 paper
and 10 characters per inch.
.ll 60
.ne 20
.h2 LINE^SPACING
.an (XSP) \%
.an (XDS) \%
Spaces between lines may be introduced by simply including a blank
or empty line (empty lines are ones that do not even contain a blank).
Alternatively the .sp (space) request may be used to force several blank
lines.
.sp 4
This line is separated from the previous one by .sp^4.
.ds
.ne 4

The .ds (double space) request has the effect of double spacing all
output, as in this sample. It remains in effect until some other
spacing is specified. The .ss (single space) request will
return you to normal mode.
.ss
.h2 PAGE^LENGTH
.an (XPL) \%
The .pl (page length) request allows you to set the number of lines per
By default .pl 66 is in effect, which works for continuous 11" computer
printed at 6^lines/inch
and also for A4 paper if you are inserting each sheet in the terminal.
:h2 CENTRING
.an (XCE) \%
Sometimes it is required to centre text in the page.
This can be achieved with the .ce (centre) request.
The following is preceeded by .ce 2.
.ce 2
This is centred.
Filling is suspended during centring.
.in 30
.sp
.ne 6
Text is always centred between the current indent and the right hand marg
Here we have .in 30 in effect, so .ce centres in positions 30-60:
.ce
like this!
.in
.h2 UNDERLINES
```

```
.an (XUL) \%
Text may be underlined just as a typist would do it,
but this is inconvenient when using most VDUs as the underline character
will obliterate the text.
There can also be problems when justification is applied to
text underlined in this way.
For example, part of this sentence was
underlined as a typist
might do it.
These problems can be avoided by underlining each word separately.
Never backspace across
a space.

It is often preferable to
.us
use the .us (underscore) request to underline all
of the next input line
as here (noting that such continuously underscored lines are
treated as a single word and cannot be justified or split
across lines) or, alternatively,
.ul 2
use the .ul (underline) request which underlines just alphanumeric charac
omitting spaces and punctuation symbols.
.h2 NEW^PAGE
.an (XR9) \%
.at (XR4)
\a.\b
.en (XR4)
ROFF will begin a new page whenever necessary.
However, you may wish to do this before the previous page is full.
The .bp (begin^page) request achieves this.

Sometimes you find that a new paragraph or section starts too near
the bottom of the page, or that a series of lines that ought to be kept
together gets split across pages.
This situation can be avoided by the .ne (need) request.
.ne^10, for example, will cause a new page to be started if there
For example

{ne 10

instructs ROFF to start a new page if
less than 10 lines are available on the present page;
otherwise it has no effect.
By default there is an implied .ne^1 after any .sp
request (or empty line) to stop a paragraph starting on the last line of
page.
.h2 SOURCE^SWITCHING
.an (XSO) \%
The .so request causes ROFF to switch temporarily to another file for
input, then return to the previous file when the other is exhausted.
It is
particularly useful for long documents that have been edited in pieces.
For example if the pieces are in files chap1, chap2, chap3, you might
```

prepare a very short file called book, containing only

```
.sp
.nf
.ne5
.li 3
.so catalog/chap1
.so catalog/chap2
.so catalog/chap3
.sp
.fi
```

then do the whole works by the simple command "roff book".
This manual was produced like this, and the file containing this chapter is actually "sourced" twice - once for the main formatted occurrence and then again to provide the "as^entered" copy in appendix^A.

```
.h2 MACRO-COMMANDS
.an (XR2) \%
.at (XR3)
\a.\b
.en (XR3)
```

Sometimes the requests necessary to set up section headers or other standard lay-outs can become quite long and may be needed many times. A very useful feature of ROFF allows the user to define his own command sequences or macro-commands and then invoke them at will. How such macro-commands are defined will be explained in section 3, but some pre-defined macros have been provided and may be used without understanding their inner workings.
This manual uses these macros extensively.

The macro definitions reside in a file LIBRARY/ROFF/DPMACROS.
To use them they must first be "sourced" with the request:

```
.li
.so LIBRARY/ROFF/DPMACROS
```

at the start of your document.
This action loads the macros ready for use and also sets up certain other options, such as the use of the
|^char`ac`ter as a pseudo blank.

```
.h3 Front^Page^Layout^Macro^^.dp
```

This macro can be used to lay out a front page like the one for this manu with only a single line of input.
The request is of the form:

```
{dp reference heading sub-heading author
```

where each of the four arguments are separated by blanks.

```
.ta 16
.in 15
.ti 0
```

reference~is the document reference number to be placed on the right hand side at the top, and on the foot of later pages.

```
.ti 0
```

```
heading~is the document name to be placed at the centre of the page.
.ti 0
sub-heading~is a sub-heading to be placed under the main heading.
.ti 0
author~is the author's name to be placed at the bottom right,
above the date.

.in 0
The front page of this manual was produced by the macro-command

{dp DP/D/30 ROFF|TEXT|FORMATTER | Tony|Voss

Note that since blanks separate the arguments any blanks within
an argument must be entered as a pseudo blank (|^in this case).
Note also that no sub-heading was required, but it was
necessary to provide a blank third argument so that the fourth
was recognised as such.
Trailing blank arguments are not needed.
.sp 2
By default the .dp macro sets up a standard IOS^Data^Processing
title.
Other departments may override this as required.
If a MIAS document were being prepared, the lines:

.nf
.an (XR8) \%
{at (dept)
MARINE|INFORMATION|&|ADVISORY|SERVICE
{en (dept)
.fi

placed
.us
before
the .dp request would cause that department to appear on the front page.
The above sequence will not be explained further here, but is covered in
section 3.

The .dp macro also arranges for the reference number, page number and cu
appear at the foot of each subsequent page.
.h3 Major^Headings^Macros^^.h1^&^.hp
The macros .h1 and .hp may be used to set up a major heading.
They take a single argument - the heading itself.
The only difference between them is that .hp always starts a new page
whereas .h1 will start on the same page if there is plenty of room.
The major heading at the top of page \(XR1) was set up with the macro-re

{hp FIRST|STEPS

Note that the section number was assigned automatically and the heading
centred, underscored and spaced above the text that follows.
When using these macros there is no need to insert blank lines yourself.
.h3 Sub-Heading^Macro^^.h2
This is similar to .h1 except that it is used to set up a level^2 or
```

sub-heading. Spacing is proportionally smaller.
A sub-section number is generated automatically.
The sub-heading on page \(XR2) was set up by the macro-request

```
{h2 MACRO-COMMANDS

.h3 Sub-Sub-Headings^Macro^^.h3
```

The .h3 macro-request generates a third level heading,
without a section number.
The heading immediately above this text was produced by the request

```
{h3 Sub-Sub-Headings|Macro||.h3

.h3 In-line^Headings^Macro^^.h4
```

The lowest level of headings supported is the underscored in-line
phrase:
.h4 In-line^heading.
This heading was generated by the macro-request:

```
{h4 In-line|heading.
```

Headings produced by .hp, .h1 and .h2 always reset the indentation
to zero and turn on filling mode.
A .h3 request places the heading against the left hand margin without
disturbing the indentation.
A .h4 heading uses the current indentation.

```
.h3 Appendix^Macro^^.ap
```

The .ap macro-request sets up an appendix, as demostrated
by the appendicies to this manual.
The format is:

```
{ap L HEADING
```

where L is the appendix letter (usually A B C etc., but could be anythin
and HEADING the heading. Note that appendicies have their pages numbered
independently from the main part of the document.

```
.h2 INDEX
```

It is possible to arrange for the automatic generation of an index
or list of contents.
How you can do this is explained in section 3.
However, if you are using the macros described in section^\(XR3)
index entries are made automatically.
The .hp, .h1, .h2, .h3 and .ap headings will be indexed,
provided you have specified an index file when you invoked ROFF.
The index file so produced contains ROFF requests and must itself
be ROFFed to get the final product.
The list of contents for this manual was produced in exactly this way.

APPENDIX B

TABLE OF REQUESTS

This table defines all the ROFF requests. Numerical values are denoted by n or +n, titles by t, and single characters by c. End labels are denoted by e and may be empty, or may have the form of a register name described in section 7. Numbers denoted +n may be preceded by one of +-*/, in which case the the previous value is increased, decreased, multiplied, or divided by n (division by zero yields zero). Otherwise the request simply replaces the value. No numeric value is allowed to be set negative; attempts to do so cause the value to be set to zero. Missing n fields are taken to be 1, missing t fields to be empty. Missing c fields turn off .cc, .hc, .ic, .pc, .sc, and .tc. Synonyms are indicated by "syn =". The entries under PAGE refer to the page number where a section discussing the request starts.

REQUEST	BREAK	DEFAULT	PAGE	MEANING
.af R f	no	f=1	22	assign format to register R, f=i,I,a,A,1,01,...
.an R +n	no	n=0	22,24	assign number to register R, R#%; if result is negative, replace by zero
.ar	no	yes		arabic page numerals (syn = .af % 1)
.at R	no		23,24,27	assign text to register R until .en (R)
.bp	yes		7,11	begin page
.br	yes		3	break
.cc c	no	c=.	19	control character is c
.ce n	yes		6	center next n text lines, break on each
.cn	no	yes	29	case normal on input
.cr	no	no	29	case reversed, exchange upper and lower case letters on input
.ds	yes	no	5	double space (syn = .ls 2)
.ef n t	no	t=''''	13	nth even page foot title is t, 1≤n≤10
.eh n t	no	t=''''	13	nth even page head title is t, 1≤n≤10
.en e			21	end all diversions labeled e, break if end of footnote
.ep	yes		11	begin an even page
.fi	yes	yes	4	fill output lines
.fn e	no		21	divert text to footnotes until .en e
.fo n t	no	t=''''	13	nth even/odd foot titles are t, 1≤n≤10

REQUEST	BREAK	DEFAULT	PAGE	MEANING
.fs t	no	t=‘‘‘‘	21	footnote separator is t
.hc c	no		16	hyphenation character is c
.he n t	no	t=‘‘‘‘	13	nth even/odd head titles are t, 1≤n≤10
.hy n	no	n=3	16	hyphenation mode is n, 0≤n≤3
.ic c	no		22,23	insertion character is c
.ig e	no		26	ignore all input until .en e
.in +n	no	n=0	4	indent left margin n spaces
.ix e	no		25	divert input to index file until .en e
.ju	yes	yes	3	justify right margin of filled lines
.li n	no		19	literal, take next n lines to be text
.ll +n	no	n=60	5	line length is n including indent
.ls +n	yes	n=1		line spacing is n
.lv n	no		13	leave n consecutive blank lines; wait until next page if necessary
.m1 +n	no	n=4	12	margin above head no. 1 is n lines
.m2 +n	no	n=2	12	margin below and including heads is n
.m3 +n	no	n=2	12	margin above and including feet is n
.m4 +n	no	n=4	12	margin below foot no. 1 is n lines
.mg n	no	empty	18	next line sets merge pattern n, 1≤n≤10
.n0	yes	yes	20	do not number output lines
.n1	yes	no	20	number output lines, reset each page
.n2	yes	no	20	number output lines, no page reset
.ne n	no		7,13	need room for n output lines with present spacing, do .bp if necessary
.nf	yes	no	4	nofill, break on each input line
.nj	yes	no	3	no right margin justification
.np n	no	no	24	no printing of output for next n pages
.nu	no	yes	20	no first character underlining
.of n t	no	t=‘‘‘‘	13	nth odd page foot title is t, 1≤n≤10
.oh n t	no	t=‘‘‘‘	13	nth odd page head title is t, 1≤n≤10
.op	yes		11	begin an odd page
.pa +n	yes	n=1	11	begin page with page number n
.pc c	no		27	parameter character is c

REQUEST	BREAK	DEFAULT	PAGE	MEANING
.pl +n	yes	n=66	6	paper length is n lines
.po +n	no	n=0	20	page offset is n, i.e. move all output n spaces right
.pr +n	no	n=0	20	print requests indented n, don't print if n ≤ line length
.ps +n	no	n=0	20	print sequence numbers of input lines indented n (n for .pr > n for .ps)
.ro	no	no		roman page numerals (syn = .af % i)
.sc c	no	c=#	29	shift character is c, 6-bit (BCD) input only
.sk +n	no		11	skip at next new page to page number n
.so c/f	no		7	insert roff source from cat/file c/f
.sp n	yes		5,13,14	insert n extra spacing lines
.ss	yes	yes		single space (syn = .ls 1)
.ta	no	all	14	tabs set by this line or next
.tc c	no		14	extra tab character is c
.ti +n	yes		4,14	temporary indent, for one line only
.tr cd...	no		18	translate c into d on output
.uf	no	no	20	underline first character of each input text line
.ul n	no		6	underline alphanumerics in next n input text lines
.us n	no		6	underline all of next n input lines
.ws n	no	n=1	14	set window suppression mode n

(In addition to the above requests, ROFF recognizes these obsolete and system debugging requests: .ab, .ad, .bm, .na, .nc, .tm, .un, .va-.vz, .v%, .v#, .ya-.yz, .y%, .y#.)

The DPMACROS are available by including them with the request

.so LIBRARY/ROFF/DPMACROS

This action sets the following control characters:
.tr ^ .ic \ .tc ~ .hc ` .pc #
and margins m1, m2, m3 and m4 all to 3.

.dp reference title sub-title author irn	set up standard document front page with the specified parameters
.hp heading	Level 1 heading on new page (indexed) - set .in 0 and .fi.
.h1 heading	Level 1 heading - can be on same page (indexed) - set .in 0 and .fi.
.h2 heading	Level 2 heading (indexed) - set .in 0 and .fi.
.h3 heading	Level 3 heading against left hand margin (indexed) - don't change .in or .fi.
.h4 heading	Level 4 heading in-line using current indentation (not indexed) - set .fi - don't change .in.
.ap L title	Set up appendix L with title

Useful registers used by the DPMACROS

REGISTER	DEFAULT	USE
(%%)	0	May be set to 2 before "sourcing" DPMACROS to indicate twin page mode.
(dept)	DATA PROCESSING	sets department on front page
(date)	current date	date for front page and footings
(ref)		document reference as used on front page and footings
(a)		last .h1 section number
(b)		last .h2 section number
(c)		last .h3 section number (not normally printed)