runoff (rf)

SYNTAX AS A COMMAND:

rf paths {-control_args}

FUNCTION: is used to type out text segments in manuscript form.

ARGUMENTS:

paths
are the pathnames of input segments or multisegment files named entryname.runoff. The runoff suffix must be the last component of each entryname; however, the suffix need not be supplied in the command line. If two or more pathnames are specified, they are treated as if runoff had been invoked separately for each one. The segments are printed in the order in which they occur in the invocation of the command.

CONTROL ARGUMENTS:
can be chosen from the following list. Any control argument specified anywhere in the command invocation applies to all segments; control arguments can be intermixed arbitrarily with segment names. Control arguments must be preceded by a minus sign.

-ball N, -bl N
converts output to a form suitable for an N typeball on a unit equipped with a selectric-type typing element. Acceptable ball numbers are 041, 012, 015, and 963. The default is the form of the terminal device being used. Use of this control argument overrides any specification set by the -device control argument (below).

-character, -ch
flags certain key characters in the output by putting the line containing the key character in a segment named entryname.chars. The normal output is not affected. Page and line numbers referring to the normal output appear with each flagged line, and reminder characters, enclosed by color-shift characters, are substituted for the key characters. The default set of key and reminder characters corresponds to those unavailable with a 963 typeball, as follows:

runoff (rf)

Key	Reminder
left square bracket	<
right square bracket	>
left brace	(
right brace	)
tilde	t
grave accent	'

The key and reminder characters can be changed by use of the .ch control line; specifying a blank reminder character removes the associated key character from the set of key characters. If a key character would print normally in the output, it should also appear in a .tr control line to turn it into a blank in the output.

-device N, -dv N
prepares output compatible with the device specified. This is usually used when the output is stored in a segment to be printed elsewhere. Suitable devices are terminals 2741, 1050, 37, and the bulk output printers, 202 or 300. Use of this control argument overrides any specification set by use of the -ball control argument; if both are used in one invocation of runoff, the last one encountered prevails.

If neither -device nor -ball is specified, the default device type is that from which the user is logged in; any unrecognized device type is assumed to support the entire ASCII character set.

-from N, -fm N
starts printing at the page numbered N. If the -page control argument is used, printing starts at the renumbered page N.

-hyphenate, -hph
When this control argument is used, a procedure named hyphenate_word_, that the user supplies, is invoked to perform hyphenation when the next word to be output does not fit in the space remaining in a line (see "Hyphenation Procedure Calling Sequence" at the end of this description). Otherwise, no attempt is made to hyphenate words.

-indent N, -in N
indents output N spaces from the left margin (default indentation is 0 except for "-device 202," which is the default for -segment and has a default indentation of 20; see also -number below). This space is in addition to whatever

indentation is established by use of the .in control word.

-no_pagination, -npgn
suppresses page breaks in the output.

-number, -nb
prints source line numbers in the left margin of the output; minimum indentation of 10 is forced.

-page N, -pg N
changes the initial page number to N. All subsequent pages are similarly renumbered. If the control line .pa is used within the segment, the -page control argument is overridden and the page is numbered according to the .pa control line.

-parameter arg, -pm arg
assigns the argument arg as a string to the internal variable "Parameter".

-pass N
processes the source segments N times to permit proper evaluation of expressions containing symbols that are defined at a subsequent point in the input. No output is produced until the last pass.

-segment, -sm
directs output to the segment or multisegment file named entryname.runout. This control argument assumes by default that the material is to be dprinted, so the segment is prepared compatible with device 202 unless another device is specified; thus, unless overridden by the -indent control argument, each printed line in the output segment is preceded by 20 leading spaces so that the text is approximately centered on the page when dprinted.

-stop, -sp
waits for a carriage return from the user before beginning typing and after each page of output (including after the last page of output).

-to N
ends printing after the page numbered N.

-wait, -wt
waits for a carriage return from the user before starting output, but not between pages.

runoff (rf)

NOTES: Output lines are built from the left margin by adding text words until no more words fit on the line; the line is then justified by inserting extra blanks to make an even right margin. Up to 20 lines each of headers and footers can be printed on each page. The pages can be numbered, lines can be centered, and equations can be formatted. Space can be allowed for diagrams. Detailed control over margins, spacing, headers, justification, numbering, and other aspects of format is provided by control lines that begin with a period. Although the control lines are interspersed within the text, they do not appear in the output segment. The output can be printed page by page to allow positioning of paper, or it can be directed into a segment. Characters not available on the device to which output is directed are replaced by blanks. If special symbols must be hand drawn, a separate segment can be created that indicates where each symbol should be placed. The user can define variables and cause expressions to be evaluated; he also has the ability to refer to (and sometimes modify) variables connected with the workings of the runoff command.

A runoff input segment contains two types of lines: control lines and text lines. A control line begins with a period; all other lines are considered text lines. A two-character control word appears in the second and third character positions of each control line. The control word can take a parameter that is separated from the control word by one or more spaces. Lines that are entirely blank are treated as if they contained a .sp 1 control line.

Text lines contain the material to be printed. If an input line is too short or too long to fill an output line, material is taken from or deferred to the next text line. A line beginning with a space is interpreted as a break in the text (e.g., the beginning of a new paragraph) and the previous line is printed as is.

Tab characters (ASCII HT) encountered in the input stream are converted to the number of spaces required to get to the next tab position (11, 21, ...). Nonprinting control characters in the input segment are discarded in the output segment. The .tr control word can be used to print these control characters in the output segment.

runoff (rf)

When an input text line ends with any of the characters ".", "?", "!", ";", or ":", or with ".", "?", or "!" followed by a double quote or ")", two blanks precede the following word (if it is placed on the same output line), instead of the normal single blank.

The maximum number of characters per input or output line is 361; this permits 120 underlined characters plus the newline character.

Terminology

The following paragraphs describe various terms that are used throughout the runoff description.

FILL AND ADJUST MODES

Two separate concepts are relevant to understanding how runoff formats output: fill mode and adjust mode. In fill mode, text is moved from line to line when the input either exceeds or cannot fill an output line. Adjust mode right justifies the text by inserting extra spaces in the output line, with successive lines being padded alternately from the right and from the left. Initial spaces on a line are not subject to adjustment. Fill mode can be used without adjust, but in order for adjust to work, fill mode must be in effect.

LINE LENGTH

The line length is the maximum number of print positions in an output line, including all spaces and indentations, but not including margins set or implied by the -device, -indent, or -number control arguments.

BREAK

A break ensures that the text that follows is not run together with the text before the break. The previous line is printed out as is, without padding.

runoff (rf)

SPACING BETWEEN LINES

Vertical spacing within the body of the text is controlled by the three control words: .ss, .ds, and .ms (for single, double, and multiple spacing respectively). Single spacing is the default. Multiple spacing is set by the control line .ms N where N-1 is the number of blank lines between text lines.

PAGE EJECT

A page eject ensures that no text after the control line causing the page eject (e.g., .bp for "begin page") is printed on the current page. The current page is finished with only footers and footnotes at the bottom, and the next text line begins the following page.

MARGINS

There are four margins on the page vertically. The first margin on the page is the number of blank lines between the top of the page and the first header; this margin is set by the .m1 control word. The second, set by .m2, specifies the number of lines between the last header and the first line of text. The third (.m3) is between the last line of text and the first footer. The fourth (.m4) is between the last footer and the bottom of the page. The default for the first and fourth margins is four lines; for the second and third, two lines.

PAGE NUMBERS

As the output is being prepared, a page number counter is kept. This counter can be incremented or set by the user. The current value of the counter can be used in a header or footer through the use of the symbol "%". A page is called odd (even) if the current value of the counter is an odd (even) number.

The page numbers can be output as either arabic (the default) or roman (using the .ro control word).

runoff (rf)

HEADERS AND FOOTERS

A header is a line printed at the top of each page. A footer is a line printed at the bottom of each page. A page can have up to 20 headers and 20 footers. Headers are numbered from the top down, footers from the bottom up. The two groups are completely independent of each other. Provision is made for different headers and footers for odd and even numbered pages. Both odd and even headers (footers) can be set together by using the .he (.fo) control words. They are set separately by using the .eh, .oh, .ef, and .of control words.

A header/footer control line has two arguments, the line number (denoted in the control line descriptions as "#"), and the title.

The line number parameter of the control line determines which header or footer line is being set. If the number is omitted, it is assumed to be 1, and all previously defined headers or footers of the type specified (odd or even) are cancelled. Once set, a line is printed on each page until reset or cancelled.

The title part of the control line begins at the first nonblank character after the line number. This character is taken to be the delimiting character, and can be any character not used in the rest of the title. If the delimiting character appears less than four times, the missing last parts of the title are taken to be blank. The three parts of the title are printed left justified, centered, and right justified, respectively, on the line. Any or all parts of the title can be null. Justification and centering of a header or footer line are derived from the line length and indentation in effect at the time of the definition of the header or footer, and are used whenever that line is output, regardless of the values at the time of use. Any occurrence of the special character "%" within a title is replaced by the current value of the page counter whenever the title is printed. To cause a percent character to be printed, "%%" must be written in the title. The special character can be changed; see the .cc control word.

runoff (rf)

Omitting the title in the control line cancels the header or footer with that number, including its space on the page (e.g., ".he 4" cancels the fourth header). A blank line in the header or footer can be achieved by a title consisting entirely of one delimiting character (e.g., ".fo 3 $" makes the third footer a blank line). Omitting both number and title of a header (footer) cancels all headers (footers) of the type specified (e.g., ".oh" cancels all headers that were specified by any .oh control line).

Expressions and Expression Evaluation

An expression can be either arithmetic or string, and consists of numbers and operators in appropriate combinations. All operations are performed in integer format, except that string comparisons are performed on the full lengths of the strings.

The order of precedence for the operators is:

^ (bit-wise negation), - (unary)
*, /, \ (remainder)
+, - (binary)
=, <, >, ≠, ≤, ≥ (all are comparison operators that yield -1 for true or 0 for false)
& (bit-wise AND)
| (bit-wise OR), ≡ (bit-wise equivalence)

Other guidelines in the use of expressions are as follows:

1) Parentheses can be used for grouping.

2) Blanks are ignored outside of constants.

3) Octal numbers consist of "#" followed by a sequence of octal digits.

4) String constants are surrounded by the double quote character; certain special characters are defined by multiple-character sequences that begin with the * character, as follows:

**	asterisk character
*"	double-quote character

*b	backspace character
*n	newline character
*t	horizontal-tab character
*s	space character
*cN	character whose decimal value is N (where N is 1 to 3 digits)

5) Concatenation of strings is performed by the juxtaposition of the strings involved, in order, left to right.

6) For positive i, k,

```
string_expression(i)
```

and:

```
string_expression(i, k)
```

are equivalent to the PL/I substr builtin function references:

```
substr(string_expression, i)
```

and:

```
substr(string_expression, i, k)
```

respectively.

7) For negative i, the substring is defined as starting -i characters from the rightmost end of the string; for negative k, the substring ends -k characters from the end of the string.

8) Evaluation of substrings takes place after any indicated concatenations; string operations have higher precedence than all the binary operations.

9) In any context other than a .sr control line or in a string comparison, a string expression is converted to an integer in such a way that a one-character string results in the ASCII numeric value of the character.

Expression evaluation takes place under the following conditions:

runoff (rf)

1) In .sr and .ts control lines.

2) In all control lines that accept an "N" or "±N" argument.

Definition and Substitution of Variables

Variables can be defined by the use of the .sr control line; their values can be retrieved thereafter by a symbolic reference. Names of the variables are composed of the uppercase and lowercase alphabetic characters, decimal digits, and "_", with a maximum length of 361 characters. When a variable is defined, it is given a type based on the type of the expression that is to be its value, either arithmetic or string. Variables that are undefined at the time of reference yield the null string. When enclosed in quotes, a null string is equivalent to an arithmetic 0. Thus, if a variable called var has not been set, this "%var%" is equivalent to 0 when used as an expression.

In substitution of variables, the name of the variable is enclosed by "%"; other occurrences of the character '%" encountered during substitution of variables are replaced by the value of the page counter; if a "%" character is to occur in the resulting output, it must be coded as "%%" (but see also the .cc control word).

Substitution of variables can occur:

1) In control lines that take an expression argument if a "%" is found as either the first or second character of the argument (substitution of variables takes place before expression evaluation).

2) In .ur control lines.

3) In all titles ('part1'part2'part3'), whether in header/footer control lines or as equation lines.

Many of the variables internal to runoff are available to the user (a complete list is given at the end of this description). These variables include control argument values

(or their defaults), values of switches and counters, and certain special functions. However, the user need not worry about naming conflicts, since an attempt to redefine an internal variable that is not explicitly modifiable is ignored; i.e., the operation of the command is unaffected.

Two special built-in symbols in runoff are provided for use in footnote and equation numbering: "Foot" contains the value of the next footnote number available (or the current footnote if referred to from within the text of the footnote) and "Eqcnt" is provided for equation numbering. The value of "Foot" is incremented by one when the closing .ft of a footnote is encountered. Any reference to "Eqcnt" provides the current value and causes its value to be incremented by one automatically; thus its value should be assigned to a variable, and the variable should then be used in all further references to that equation number.

Default Conditions

When no control words are specified, runoff prints the text single spaced, right adjusted, with no headers, no footers, and no page numbers.

If page numbers are substituted in headers or equations, they are arabic.

A page consists of 66 lines, numbered 1 through 66. The first line is printed on line 7, and the last on line 60, if no headers or footers are used. If headers are used, there are four lines of top margin (.m1 4), the headers, two blank lines (.m2 2), and then the text. If footers are used, there are two lines skipped after the text (.m3 2), footers printed, and four lines of bottom margin (.m4 4).

A line is 65 characters long; the left margin is that of the typewriter. The output is compatible with whatever is normal for the device from which the runoff command is executed. The entire segment is printed, with no wait before beginning or between pages.

runoff (rf)

NOTES ON CONTROL LINE FORMATS: The following discussion gives a description of each of the control words that can be interspersed with the text for format control. Control lines do not cause an automatic break unless otherwise specified. Arguments of the control words are in the following form:

#	integer constant
N	integer expression
+N	integer expression preceded by optional + or - sign
<expression>	arbitrary expression (string or integer)
c	character
cd	character pair
f	segment name
'part1'part2'part3'	a title whose parts are to be left justified, centered, and right justified respectively.

.ad
Adjust: text is printed right justified. Fill mode must be in effect for right justification to occur. Fill mode and adjust mode are the default conditions. This control line causes a break.

.ar
Arabic numerals: when page numbers (% variable) are substituted into text or control lines as a result of a .ur control line or into a title or equation as it is printed, they are in arabic notation. This is the default condition.

.bp
Begin page: the next line of text begins on a new page of output. This control line causes a break.

.br
Break: the current output line is finished as is, and the next text line begins on a new output line.

.cc c
Control character: this control line changes the character used to surround the names of symbolic variables when they are referenced to c. The default special character is "%". The character specified by c must thereafter be used to

refer to symbolic variables, while percent signs are treated literally. Either ".cc %" or .cc restores the percent sign as the special character.

.ce N
Center: the next N text lines are centered. Control lines and blank lines are not counted as part of the N lines being centered. If N is missing, 1 is assumed. This control line implies ".ne N" (or ".ne 2N" if double spacing) so that all lines centered are on the same page. A break occurs.

.ch cd..
Characters: each occurrence of the character c is replaced in the chars segment (the segment named entryname.chars) by the character d, set off by color-shift characters. If the d character is blank, or an unpaired c character appears at the end of the line, the c character is not flagged; it either occurs as itself in the chars segment or not at all if no other character on the line was flagged.

.ds
Double space: begin double spacing the text. This control line causes a break.

.ef # 'part1'part2'part3'
Even footer: this defines even page footer line number #. If # is omitted, 1 is assumed. If both # and the title (parts 1 to 3) are omitted, all even footers defined by any .ef control line are cancelled. For more information, see the previous discussion entitled "Headers and Footers."

.eh # 'part1'part2'part3'
Even header: this defines even page header line number #. If # is omitted, 1 is assumed. If both # and the title (parts 1 to 3) are omitted, all even headers defined by any .eh control line are cancelled. For more information, see the previous discussion entitled "Headers and Footers."

.eq N
Equation: the next N text lines are taken to be equations. If N is missing, 1 is assumed. This control line implies ".ne N" (or ".ne 2N" if double spacing) so that all equations are on the same page. The format of the equations should be 'part1'part2'part3' just as in headers and footers.

runoff (rf)

.ex text
Execute: the remainder of the control line (text) is passed to the Multics command processor. Substitution of variables can occur if the first or second character of text is "%".

.fh 'part1'part2'part3'
Footnote header: before footnotes are printed, a demarcation line is printed to separate them from the text. The format of this line can be specified through the title in the .fh control line. This title is printed in the same manner as headers/footers and equations. The default footnote header is a line of underscores from column one to the right margin.

.fi
Fill: this control line sets the fill mode. In fill mode, text is moved from line to line to even the right margin, but blanks are not padded to justify exactly. Fill mode is the default condition. This control line causes a break.

.fo # 'part1'part2'part3'
Footer: even and odd footers are set at the same time; this is equivalent to:

```
.ef # 'part1'part2'part3'
.of # 'part1'part2'part3'
```

If # is omitted, 1 is assumed. If both # and the title (parts 1 to 3) are omitted, all footers are cancelled. For more information, see the discussion entitled "Headers and Footers."

.fr c
Footnote reset: this control line controls footnote numbering according to the argument c. Permitted values of c are:

t Footnote counter is reset at the top of each page. This is the default condition.
f Footnote counter runs continuously through the text.
u Suppress numbering on the next footnote.

.ft
Footnote: when .ft is encountered, all subsequent text until the next .ft line is treated as a footnote. Any further text on the .ft line is ignored. If a footnote occurring near the bottom of a page does not fit on the

page, as much as necessary is continued at the bottom of the next page. If a footnote reference occurs in the bottom or next to bottom line of a page, the current page is terminated and the line with the footnote reference is printed at the top of the next text page.

.gb STR
Go back: the current input segment is searched from the beginning until a line of the form ".la STR" is found; "STR" in this case means "the rest of the line." Processing is continued from that point.

.gf STR
Go forward: same as .gb, except search forward from the current position in the input segment.

.he # 'part1'part2'part3'
Header: even and odd headers are set at the same time. This is equivalent to:

```
.eh # 'part1'part2'part3'
.oh # 'part1'part2'part3'
```

If # is omitted, 1 is assumed. If both # and the title (parts 1 to 3) are omitted, all headers are cancelled. For more information, see the discussion entitled "Headers and Footers."

.if f <expression>
Insert file: the segment specified by f is inserted into the text at the point of the ".if f" control line. The inserted segment can contain both text and control lines. No break occurs. The effect is as if the control line were replaced by the segment. Inserts can be nested to a maximum depth of 30. The argument f is the entryname of a runoff input segment. If the runoff suffix is not included in the .if line, it is supplied. The input file is located by use of the translator search list that has the synonym trans. For more information on search lists, see the search facility commands and, in particular, the add_search_paths command description in this manual. If a second argument is provided, it is evaluated in the same fashion as the expression in .sr, and its value and type are associated with the identifier "Parameter"; if no second argument is provided the value of "Parameter" remains unchanged (or undefined). (In either case, the prior value of "Parameter" is not pushed down).

runoff (rf)

.in ±N
Indent: the left margin is indented N spaces by padding N leading spaces on each line. The right margin remains unchanged. By default N is 0. The margin can be reset with another ".in N" request. Either .in or ".in 0" resets the original margin. If N is preceded by a plus or a minus sign, the indentation is changed by N rather than reset. This control line causes a break.

.la STR
Label: defines the label STR for use as the target of the .gb or .gf control word.

.li N
Literal: this request causes the next N lines to be treated as text, even if they begin with a period (.). If N is not specified, 1 is assumed.

.ll ±N
Line length: the line length is set to N. The left margin stays the same, and no break occurs. If N is not specified, 65 is assumed. If N is preceded by a plus or a minus sign, the line length is changed by N rather than reset.

.ma ±N
Margins: top and bottom margins are set to N lines. If N is preceded by a plus or a minus sign, the margin is changed by N rather than reset. The margin is the number of lines printed above the first header and below the last footer. If N is not specified, 4 is assumed. This control line is equivalent to:

.m1 ±N
.m4 ±N

Note: Care should be taken in using a top or bottom margin of less than three lines if output is to be directed to an off-line printer (-sm). Such printers typically are set up to skip automatically the first and last 3 lines on each page. Runoff takes this into account by putting out fewer newlines to compensate for the printer; the compensation cannot work for .m1 or .m4 less than 3. It is possible to get around this problem by using the -device 037 control argument when invoking runoff and special control arguments to the

command that set up requests for the off-line printer (dprint).

.mp +N
Multiple pages: format the output text so that it prints on every Nth page (i.e., skips N-1 blank sheets of paper between printed pages). This control line is valid cnly for output intended for the bulk printer. If N is not specified, 1 is assumed.

.ms +N
Multiple space: begin multiple spacing text, leaving (N-1) blank lines between text lines. If N is preceded by a plus or a minus sign, the spacing is changed by N rather than reset. If N is not specified, 1 is assumed. This control line causes a break.

.m1 +N
Margin 1: the margin between the top of the page and the first header is set to N lines, or changed by N if N is signed. If N is not specified, 4 is assumed. See note in description of .ma.

.m2 +N
Margin 2: the number of blank lines between the last header and the first line of text is set to N, or changed by N if N is signed. If N is not specified, 2 is assumed.

.m3 +N
Margin 3: the number of blank lines printed between the last line of text and the first footer is set to N, or changed by N if N is signed. If N is not specified, 2 is assumed.

.m4 +N
Margin 4: the margin between the last footer and the bottom of the page is set to N lines, or changed by N if N is signed. If N is not specified, 4 is assumed. See note in description of .ma.

.na
No adjust: the right margin is not adjusted. This does not affect fill mode; text is still moved from one line to another. This control line causes a break.

.ne N
Need: a block of N lines is needed. If N or more lines

runoff (rf)

remain on the current page, text continues as before; otherwise, the current page is ejected and text continued on the next page. The number of lines remaining is calculated by subtracting from the current page length the sum of the number of lines already printed and the number of lines reserved for footers, footnotes, and bottom margins. No break is implied; if a line is partially formatted but not yet printed when the .ne is encountered, it is ignored in the calculation of lines remaining (i.e., it is neither printed nor in possession of reserved space). Similarly, a footnote or footer defined after the .ne does not have space reserved at the time the .ne is encountered. If N is not specified, 1 is assumed. If several .ne control lines occur consecutively, the N's are not added together; only the largest N has effect.

.nf
No fill: fill mode is suppressed, so that a break is caused after each text line. Text is printed exactly as it is in the input segment. This control line causes a break.

.of # 'part1'part2'part3'
Oddfooter: this defines odd page footer line number #. If # is omitted, 1 is assumed. If both # and the title (parts 1 to 3) are omitted, all footers defined by any .of control line are cancelled. For more information, see the discussion entitled "Headers and Footers."

.oh # 'part1'part2'part3'
Oddheader: this defines odd page header line number #. If # is omitted, 1 is assumed. If both # and the title (parts 1 to 3) are omitted, all headers defined by any .oh control line are cancelled. For more information, see the discussion entitled "Headers and Footers."

.op
Odd page: the next page number is forced to be odd by adding 1 to the page number counter if necessary. A break is caused and the current page is ejected. No blank even page is made; the even page number is merely skipped.

.pa +N
the current line is finished as is (i.e., a break occurs) and the current page is ejected. The page number counter is set to N, or is changed by N if N was signed. If N is omitted, the page number counter is incremented by 1.

runoff (rf)

.pi N
Picture: if N lines remain on the present page, N lines are spaced over; otherwise, the text continues as before until the bottom of the page is reached, N lines are skipped on the next page before any text is printed. Headers are printed normally; the space resolved is below the headers. This option can be used to allow for pictures and diagrams. If several .pi control lines occur consecutively, each N is added to the number of lines pending and the total is checked against the space remaining on the page. All pending space is allotted together. If the total is greater than the usable space on a page, the next page contains only headers and footers and the rest of the space is left on the following page. If N is not specified, 1 is assumed.

.pl ±N
Page length: the page length is set to N lines. If N is not specified, 66 is assumed. If N is preceded by a plus or a minus sign, the page length is changed by N rather than reset.

.rd
Read: one line of input is read from the user input I/O switch; this input line is then processed as if it had been encountered instead of the .rd control line. Thus it can be either a text line or a control line; a break occurs only if the replacement line is one that would cause a break.

.ro
Roman numerals: when page numbers (% variable) are substituted into text or control lines as a result of a .ur control line or into a title or equation as it is printed, they are in lowercase roman notation. This can be reset to arabic numerals (the default) by use of the .ar control line.

.rt
Return: cease processing characters from the current input segment. If the current input segment was entered by a .if control line in another segment, return to the line following the .if control line.

.sk N
Skip: N page numbers are skipped before the next new page by adding N to the current page number counter. No break in text occurs. This control line can be used to leave out a page number for a figure. If N is not specified, 1 is assumed.

.sp N
Space N lines: If N is not specified, 1 is assumed. If not enough lines remain on the current page, footers are printed and the page ejected, but the remaining space is not carried over to the next page. The N blank lines are produced in addition to any that may occur automatically due to a .ds or .ms control line. For example, if .sp 4 is used with .ss or .ms 1, in effect four blank lines appear between two text lines, with .ds or .ms 2, five lines appear, with .ms 3, six lines.

After skipping the space, the equivalent of a .ne 2 is performed in an attempt to avoid separating the first line of a paragraph at the bottom of a page from the rest of the paragraph on the next page. The .ne feature can be avoided, if the user so desires, by using a blank line rather than .sp. Otherwise, a blank line is treated as if it were a .sp 1 control line.

This control line causes a break.

Note: A series of .sp control lines such as:

```
.sp a
.sp b
```

is not always equivalent to a single .sp control line whose argument is the sum of the individual arguments:

```
.sp a+b
```

If the .sp a finishes a page, causing a page ejection, b blank lines are produced at the top of the new page. If .sp a+b is used, the space does not appear at the top of the next page.

.sr name <expression>
Set reference: associates value of <expression> with the identifier name. The type of name is set to the type of <expression> (either numeric or string); if the expression is not provided or cannot be properly evaluated, a diagnostic

message is printed. The name identifier can be either a user-defined identifier or one of the built-in symbols that the user can set (see "Built-In Symbols" below).

.ss
Single space: begin single spacing text. This is the default condition. This control line causes a break.

.tr cd..
Translate: the nonblank character c is translated to d in the output. An arbitrary number of cd pairs can follow the initial pair on the same line without intervening spaces. An unpaired c character at the end of a line translates to a blank character. (Translation of a graphic character to a blank only in the output is useful for preserving the identity of a particular string of characters, so that the string is neither split across a line, nor has padding inserted within it.) If several .tr control lines are used in a segment, the cd pairs are "added together." Also a particular c character can be translated to a different d character by using a new .tr control line to override the previous translation. To cancel a cd pair (i.e., have the c character print out as itself), use another .tr control line of the form ".tr cc". A .tr control line with no cd pair is ignored.

.ts N
Test: process the next input line if the value of N does not equal zero (false). If N is not specified, 1 is assumed.

.ty STR
Type: write STR (i.e., the rest of the control line) onto the error output I/O switch. Substitution of variables can occur if the first or second character of STR is "%". If STR is omitted, a blank line is written onto the I/O switch.

.un N
Undent: the next output line is indented N spaces less than the current indentation. Adjustment, if in effect, occurs only on that part of the line between the normal left indentation and the right margin. If N is not specified, its value is the current indentation value (i.e., the next output line begins at the current left margin). This control line causes a break.

runoff (rf)

.ur text
Use reference: the remainder of the .ur control line (text) is scanned, with variables of the form "%name%" replaced by their corresponding values (converted back to character string form if they were numeric). The line thus constructed is then processed as if it had been encountered in the original input stream (e.g., it can be another control line, including possibly another .ur).

.wt
Wait: read one line from the user_input I/O switch and discard it (see the .rd control word description).

.*
This line is treated as a comment and ignored. No break occurs.

.~
This line is treated as a comment and ignored with respect to the output segment. However, the line is printed in the appropriate place in the chars output segment.

Summary of Control Arguments

-ball N, -bl N
Convert output to a form suitable for an N typeball.

-character, -ch
Create entryname.chars, listing page and line numbers with red reminder characters where certain characters, normally not printable, must be drawn in by hand.

-device N, -dv N
Prepare output compatible with device N.

-from N, -fm N
Start printing at the page numbered N.

-hyphenate, -hph
Call user-supplied procedure to perform hyphenation.

-indent N, -in N
Set initial indentation to N.

runoff (rf)

-no_pagination, -npgn
Suppress page breaks.

-number, -nb
Print source segment line numbers in output.

-page N, -pg N
Change the initial page number to N.

-parameter arg, -pm arg
Assign arg as a string to the internal variable "Parameter".

-pass N
Make N passes over the input.

-segment, -sm
Direct output to the segment or multisegment file named entryname.runout, where entryname is the name of the input segment.

-stop, -sp
Wait for a carriage return before each page.

-to N
Finish printing after the page numbered N.

-wait, -wt
Wait for a carriage return before the first page.

Summary of Control Words

The following conventions are used to specify arguments of control words:

#	integer constant
c	character
cd	character pair
exp	expression (either numeric or string)
N	integer expression
±N	± indicates update by N; if sign not present, set to N
f	segment name
t	title of the form 'part1'part2'part3'

runoff (rf)

Request	Break	Default	Meaning
.ad	yes	on	Right justify text
.ar	no	arabic	Arabic page numbers
.bp	yes		Begin new page
.br	yes		Break, begin new line
.cc c	no	%	Change special character from % to c
.ce N	yes	N=1	Center next N lines
.ch cd....	no		Note "c" in chars segment as "d"
.ds	yes	off	Double space
.ef # t	no		Defines even footer line #
.eh # t	no		Defines even header line #
.eq N	yes	N=1	Next N lines are equations
.ex text	no		Call command processor with "text"
.fh t	no	line of underscores	Format of footnote demarcation line
.fi	yes	on	Fill output lines
.fo # t	no		Equivalent to: .ef # t .of # t
.fr c	no	t	Control footnote numbering: "t" reset each page "f" continuous "u" numbering suppressed for next footnote
.ft	no		Delimits footnotes
.gb STR	no		"go back" to label STR
.gf STR	no		"go forward" to label STR
.he # t	no		Equivalent to: .eh # t .oh # t
.if f exp	no		Segment f.runoff inserted at point of request; value of "exp" assigned to "Paramater"
.in ±N	yes	N=0	Indent left margin N spaces
.la STR	no		Define label STR
.li N	no	N=1	Next N lines treated as text
.li ±N	no	N=65	Line length is N
.ma ±N	no	N=4	Equivalent to: .m1 ±N .m4 ±N

runoff (rf)

Request	Break	Default	Meaning
.mp ±N	no	N=1	Print only every N-th page
.ms ±N	yes	N=1	Multiple space N lines
.m1 ±N	no	N=4	Margin above headers set to N
.m2 ±N	no	N=2	Margin between headers and tect set to N
.m3 ±N	no	N=2	Margin between text and footers set to N
.m4 ±N	no	N=4	Margin below footers set to N
.na	yes	off	Do not right justify
.ne N	no	N=1	Need N lines; begin new page if not enough remain
.nf	yes	off	Do not fill output lines; print them exactly as entered
.of # t	no		Defines odd footer line #
.oh # t	no		Defines odd header line #
.op	yes		Next page number is odd
.pa ±N	yes		Begin page N
.pi N	no	N=1	Skip N lines if N remain; otherwise skip N lines on next page before any text
.pl ±N	no	N=66	Page length is N
.rd	no		Read one line of text from the user_input I/O switch and process it in place of .rd line
.ro	no	arabic	Roman numeral page numbers
.rt	no		"Return" from this input segment
.sk N	no	N=1	Skip N page numbers before next new page
.sp N	yes	N=1	Space N lines
.sr sym exp	no		Assign value of "exp" to variable named "sym"
.ss	yes	no	Single space
.tr cd....	no		Translate nonblank character c into d on output
.ts N	no	N=1	Process the next input line only if N is not zero
.ty STR	no		Write "STR" onto the error_output I/O switch
.un N	yes	left margin	Indent next text line N spaces less
.ur text	no		Substitute values of variables in "text", and scan the line

runoff (rf)

Request	Break	Default	Meaning
.wt	no		again Read one line of text from the user_input I/O switch and discard it (for synchronization with terminal)
.*	no		Comment line; ignored
.~	no		Comment line; ignored

Built-in Symbols

Only those symbols marked yes in the Set column can have values assigned by the user.

All symbols are of type Number unless they are specified to be of type String.

Control words and control arguments that affect the values of the variables are indicated in parentheses: (x/y) indicates that x sets the switch to true (-1), and y sets it false (0); (a) or (a, b, c) indicates that it is affected by a or by a, b and c.

Symbol	Set	Value
Ad		Adjust (.ad/.na)
Ce		Number of lines remaining to be centered (.ce)
CharsTable	yes	Translation table for chars segment output (String) (.ch)
Charsw	yes	A chars segment is being created (-character)
ConvTable	yes	Translation table for output. Product of DeviceTable and TrTable (String) (.tr, -device)
Date		Date of this invocation of runoff; format is mm/dd/yy (String)
Device	yes	Type of device output is to be formatted for (-device, -ball, -segment)

runoff (rf)

Symbol	Set	Value
DeviceTable	yes	Translation table for physical device (String) (-device)
Eq		Equation line counter (.eq)
Eqcnt	yes	Equation reference counter (incremented each reference)
ExtraMargin	yes	Indent entire text this many spaces (-segment, -device, -indent)
Fi		Fill switch (.fi/.nf)
FileName		Name of current primary input segment (String)
Filesw		True if output is going to a segment (-segment)
Foot	yes	Footnote counter (.ft, .fr)
FootRef	yes	Footnote reference string in footnote body (String)
Fp	yes	First page to print (set at the beginning of each pass to the value of From)
Fr		Footnote counter reset switch
From	yes	First page to print (-from)
Ft		Footnote processing switch (.ft)
Hyphenating	yes	True if an attempt to break a word should be made (-hyphenate)
In		Indent to here (.in)
InputFileName		Name of current input segment (String) (.if)
InputLines		Current line number in current source file
LinesLeft		Number of usable text lines left on this page
Ll		Line length (.ll)
Lp	yes	Last page to print (initialized each pass from To)
Ma1		Space above header (.ma, .m1)
Ma2		Space below header (.m2)
Ma3		Space above foot (.m3)
Ma4		Space below foot (.ma, .m4)
Ms		Spacing between lines (ss = 1, ds = 2, etc.) (.ms, .ss, .ds)
MultiplePagecount		Form feeds between pages to printer (.mp)
NestingDepth		Index into stack of input files (.if)

runoff (rf)

Symbol	Set	Value
Nl		Last used line number
NNp	yes	Next page number (-page, .pa)
NoFtNo		True to suppress number on next footnote reference (.fr)
NoPaging	yes	True if no pagination is desired (-no_pagination)
Np	yes	Current page number (.pa, -page, initialized each pass from Start)
PadLeft		Alternate left/right padding switch (.un, .ad)
Parameter	yes	Argument passed during insert processing (-parameter, .if)
Passes	yes	Number of passes left to make (= 1 when printing is being performed) (-pass)
Pi		Space needed for pictures (.pi)
Pl		Page length (.pl)
Print	yes	Whether or not to print ((Fp $\leq$ Np $\leq$ Lp) & (Passes $\leq$ 1))
Printersw		Output is intended for bulk printer (-device, -segment)
PrintLineNumbers	yes	True if source line numbers are to be printed in output (-number)
Roman		Roman numeral pagination (.ro/.ar)
Selsw		True if typeball other than 963 is being used (-ball)
Start	yes	Initial page number (-page)
Stopsw	yes	Stop between pages of output (-stop)
TextRef	yes	Footnote reference string in main text (String)
Time		Local time, in seconds, since January 1, 1901.
To	yes	Last page to be printed (-to)
TrTable	yes	Translation table for user-supplied substitutions (String) (.tr)
Un		Undent to here (.un)
Waitsw	yes	Wait for input before printing first page (-wait)

runoff (rf)

NOTES ON HYPHENATION PROCEDURE CALLING SEQUENCE: The runoff command provides a means whereby a user-supplied program can be called whenever the space available on a line is less than the length of the next word (including attached punctuation, if any). The mechanism is activated by use of the -hyphenate control argument, and the PL/I calling sequence is provided below.

```
declare hyphenate_word_ entry(char(*) unaligned, fixed bin,
   fixed bin);

call hyphenate_word_(string, space, break);
```

LIST OF HYPHENATION CONTROL ARGUMENTS:

string
is the text word that is to be split. (Input)

space
is the number of print positions remaining in the line. (Input)

break
is the number of characters from the word that should be placed on the current line; it should be at least one less than the value of space (to allow for the hyphen), and can be 0 to specify that the word is not to be broken. Thus if the word "calling" is to be split, and 6 spaces remain in the line, the procedure should return the value 4 (adjustment is performed after hyphenation). (Output)

EXAMPLES: The following pages show the creation of a runoff segment and the result of invoking the runoff command on that segment. For an explanation of any of the control lines, refer to the respective control word definition earlier in this command description. Particularly notice the following:

runoff (rf)

1) The line length control is given before any headers and footers. A user who wants a line length other than the default one specifies it before specifying his headers and footers; if the user does not, the headers and footers on the first page are formatted for the default line length.

2) The .sr control line associates the page number count, at the time the title "RUNOFF SAMPLE PAGE" is printed, with the identifier rfsample. Refer to the last line of the segment (the .ur control line) to see how this reference is used.

3) The translate character (!) is used both to "count" spaces (see the a, b, and c items of 2, below) and to prevent an unattractive line split (see the last line of the segment.

```
qedx
a
.pl 84
.ll 80
.tr !
.fo 1 $$3-%$AG92$
.fo 2 $
.fo 3 $runoff sample  page$runoff sample page$runoff sample
page$
.brf
.bbl 1
.he "_____""_____"
.he 2 $
.he 3 XrunoffXXrunoffX
.he 4 8_____88_____8
.m1 6
.m2 3
.m3 2
.m4 6
.sp 7
.ce
RUNOFF SAMPLE PAGE
.sr rfsample %
.sp 2
.inl 0
```

runoff (rf)

The runoff command lets the user format his text segments through a variety of control words. The control words specify such things as:

```
.sp 2
.in 10
.un 5
```

1. Page length and line length (.pl and .ll respectively). If not specified by the user, these control word are given default values of:

```
.sp
.in +5
.li
.pl 66
.br
.li
.ll 65
.in -5
.sp
.un 5
```

2. Headers and footers, for all pages or for just odd numbered or just even numbered pages. The control words for headers and footers are as follows:

```
.sp
.in +5
.un 5
```

a.!!!Headers and footers on both odd and even numbered pages (.he and .fo)

```
.sp 1
.un 5
```

b.!!!Header and footers on just odd numbered pages (.oh and .of)

```
.sp 1
.un 5
```

c.!!!Headers and footers on just even numbered pages (.eh and .ef)

```
.sp 1
.in -5
.un 5
```

3. Margins that control vertical spacing in relation to the top of the page, headers, text, footers, and the bottom of the page. These margins are defined as follows:

```
.sp
.in +5
```

runoff (rf)

```
.un 5
a.   Between top of page and first header (.m1)
.spb
.un 5
.bp
b.   Between last header and first line of text (.m2)
.sp
.un 5
c.   Between last line of text and first footer (.m3)
.sp
.un 5
d.   Between the last footer and bottom of page (.m4)
.in -5
.sp
If not specified by the user, these margins are given
default values of:
.sp
.in +5
.nf
.li 4
.m1 4
.m2 2
.m3 2
.m4 4
.fi
.in 0
.sp 2
     To see the runoff segment that created this page, see
the preceding pages.
\f
w example.runoff
q
<ready message>

rf example -wt
```

runoff (rf)

RUNOFF SAMPLE PAGE

The runoff command lets the user format his text segments through a variety of control words. The control words specify such things as:

1. Page length and line length (.pl and .ll respectively). If not specified by the user, these control word are given default values of:

   ```
   .pl 66
   .ll 65
   ```

2. Headers and footers, for all pages or for just odd numbered or just even numbered pages. The control words for headers and footers are as follows:

   a. Headers and footers on both odd and even numbered pages (.he and .fo) spf
   b. Header and footers on just odd numbered pages (.oh and .of) spf
   c. Headers and footers on just even numbered pages (.eh and .ef) spf

3. Margins that control vertical spacing in relation to the top of the page, headers, text, footers, and the bottom of the page. These margins are defined as follows:

   a. Between top of page and first header (.m1)

   b. Between last header and first line of text (.m2)

   c. Between last line of text and first footer (.m3)

   d. Between the last footer and bottom of page (.m4)

   If not specified by the user, these margins are given default values of:

   ```
   .m1 4
   .m2 2
   .m3 2
   .m4 4
   ```

To see the runoff segment that created this page, see the preceding pages.