Roach · IPC

MULTICS TECHNICAL BULLETIN MTB 337

To: MTB Distribution

From: E. J. Wallman

Date: 1977 May 9

Subject: New runoff implementation

This document describes a new implementation of runoff intended for release with MR 6.0.

Comments may be returned via system mail to Wallman.Multics on System M or regular mail to Ed Wallman, MailDrop K-28, Phoenix.

INTRODUCTION

It has long been recognized that there are several major deficiencies in the current implementation of runoff. Not the least among these are:

- The implementation is done in a language (BCPL) not part of the standard Multics product and not supported by Honeywell.
- The available expertise in the implementation language is so thin that it is very difficult to schedule timely responses to required bug fixes and virtually impossible to obtain commitments for desirable enhancements.
- The formatting algorithms used do not lend themselves readily to extension to sophisticated techniques of document production (multi-column text, tabular data, insertion of line art and graphics, etc.).
- The implementation does not lend itself to extension for support of modern document transcription devices such as Diablo (1) printer terminals and photocomposing machines.

---

(1) Registered Trademark, Xerox Corporation.

When considered in the light of high levels of interest on the part of various current and prospective customers, these (and similar) deficiencies led to the conclusion that a new, more flexible and extensible implementation of runoff was needed. Hence, an implementation effort was begun in Phoenix embodying advanced text formatting algorithms and a high capability for extensibility. This development has now reached a point where it may be considered as a replacement for the BCPL implementaion.

HIGHLIGHTS OF THE IMPLEMENTATION

- A new control structure and syntax is provided. The new controls have much more mnemonic content and generally occur in matched pairs (e.g., "in"/"out", "on"/"off", "begin"/"end", "left"/"right", etc.). The existing BCPL controls are mapped onto a subset of the new controls.

- Input is taken from single segment files treated as continuous character strings with lengths derived from the bitcounts of the segments. The ability to specify up to 100 input files per command invocation is retained. Output is written to I/O switches attached through the tty_ or vfile_ I/O modules, thus permitting output files to grow to multi-segment files for very large documents.

- The assumed default device for terminal output is the terminal type recorded in the user's login data. The assumed default device for file output is an ASCII line printer.

- All command control arguments are retained and they perform the same (or equivalent) functions (see discussion of -ball under INCOMPATIBILITIES following). Several new control arguments are provided. The undocumented "^<name>" construct implying negation of switch-type control arguments is not supported.

- Formatting is accomplished by means of text blocks. Formatted lines of text obtained from the input are accumulated until an explicit or implied text break; then the block is composed into the output page. If the block may be split between pages, a minimum number of lines (long known in the typesetting industry as a "widow") must appear on each page. The widow size is controllable by the user. Most blocks are splittable, but certain ones (e.g., equation blocks, picture blocks, "keep" blocks) are not. Most blocks may appear anywhere on the output page and are composed as they occur. Certain blocks (e.g., header blocks, footer blocks, named blocks) either appear at fixed locations on the page or at a location specified by the user.

- The characteristics of the device being used for transcription are given in an external device driver table. The use of this driving table technique greatly eases the problem of supporting new devices as they become available and allows the program code (at least as a goal) to be device independent.

- The implementation is stingy with CPU time without regard to memory usage, reflecting the relative cost of these resources. Testing has shown more than 40% less CPU time, somewhat more memory usage, and more page faults than the BCPL implementation for a set of more than 50 input test files.

  Continuing development will address the issue of optimizing system resource usage, assuring a minimum dollar cost.

- The implementation makes extensive use of temporary segments (and the temporary segment manager) instead of allocations in free storage or in the stack. This technique allows much more information to be retained during an invocation. For example, insert file data is retained in a temporary segment and sufficient space is available to allow the implementation to "know" over 900 insert files. Part of the insert file data retained is the <name> and location of every .la control encountered. This retained data makes it unnecessary to ever rescan an input file when searching for labels. In support of dynamic insert files (such as generated by the runoff_fcn program), the bitcount and date-time-modified are also retained and checked on every file insertion.

- Active functions are directly supported by treating them as pseudo-variables during variable substitution. The same active function syntax as that supported by the command processor is used and the bracketed string is enclosed within the special delimiter ("%") characters. For example, if the author of a document expected a response from the addressees within a specified time limit, say two weeks from next Tuesday, the string:

  .ur Your response is expected by %[long_date Tue 2 weeks]%.

  would generate an appropriate sentence.

INCOMPATIBILITIES

The following points are presented as actual and suspected incompatibilities between this implementation and the BCPL implementation. Most involve the placement of text on the page and the visual aspect of the composed output.

- Text lines are padded for alignment by attempting to distribute white space uniformly across the line instead of by the alternating left-right method.

  This algorithm more closely approximates the uniform proportional expansion technique used in classical typesetting. Moreover, it allows extention of the code to support devices with the capability in a more straightforward manner.

- Multiple isolated occurrences of the special delimiter character ("%") do not yield multiple references to the value of the internal page counter.

  The conventional reference to the value of the page counter has been redefined as a single, unpaired occurrence of the special delimiter character ("%") instead of isolated occurrences. Hence, the control:

  .fo ""page % of %""

  such as used in the automated MCR generator will not produce the desired result. This construct must be rewritten as:

  .fo ""page %Np% of %Np%""

  Note that the parsing algorithm for the special delimiter has not been changed other than the redefinition of this one exceptional case.

- The "-nopagination" control argument is retained but is mapped onto the "-galley" control argument.

  Galley format is used traditionally in the printing industry to check grammar, spelling, and techincal content of text before it is composed into pages. The format is defined as unpaged, single column text without running headers and footers and with footnotes immediately following the referencing paragraph.

  Note that the only real difference between the implementations lies in the display of headers and footers, hence, the use of -galley for info segs will produce identical results.

- The value displayed for the -number control argument is the input file line number of that line containing the first printed character of the output line.

  This item has long been a sore point since the BCPL implementation displays the line number of the input file line that caused the line to be printed. This line may be nowhere near the line that contains the string printed, as witness

header and footer lines in -number mode. Lines numbers are not displayed on inserted blank lines.

- The label names in a file must be unique.

  The retention of label information from files in order to reduce file searching precludes non-unique label names.

DETAILED DIFFERENCES

This section gives all the known differences between the BCPL implementation and the proposed one.

Missing:

- ^<name> negation of switch control args is not supported.
- An input file may not be an MSF.

Changed:

- New controls are proposed. See mapping below and control descriptions in the attached MPM documentation.
- Line padding is distributed.
- The -ball <n> control argument is mapped into -device 2741-<n>.
- Only one single "%" may be used for the page counter in use-ref lines and title parts.
- Label names must be unique within an input file.
- The PadLeft builtin has a fixed value of 0.
- The number displayed for the -number option refers to the input line containing the first character on the output line.
- The maximum number of characters in a line has been raised from 361 to 1800.
- The -hyphenate control arg takes an optional value specifying the smallest separated word part.
- Three different breaks are defined; format, block, and page. Most controls that break cause format breaks, some cause block breaks, and a few cause page breaks.

- Picture blocks are not split. This means that three stacked pictures, each 2/3 page, will occupy three pages in the output, not two pages. Picture blocks are actually a special form of keep blocks that permit main line text to be promoted ahead of them.

Added:

- Automatic widow processing.
- Direct support of active functions.
- Controls embedded in text (using .ur controls).
- Builtin artwork and math symbols.
- New control arguments:

-linespace	change default linespace value
-galley	unpaged output for proofing
-lines	to inspect format of small parts of doc
-noart	to inhibit artwork generation

- Documentation of the -no_control, -no_fill, and -file control args.
- Additional control functions.

  text alignment at right margin, page center, inside and outside margins.

  user defined counters

  text titles

  keep blocks

  named blocks

  localized hyphenation control

  control over widowing

MAPPING OF OLD/NEW CONTROLS

The mapping of the old controls onto the new is as follows:

.*	.*
.~	.~
.ad	.alb
.ar	.srm ar
.bp	.brp
.br	.brf
.cc	.cc
.ce	.bbc n
.ch	.ch
.ds	.ls 2
.ef	.foe
.eh	.hee
.eq	.bbe n
.ex	.ex
.fh	.hef
.fi	.fin
.fo	.foa
.fr t	.ftr
.fr f	.fth
.fr u	.ftu
.ft	.bbf (odd occurrences)
	.bef (even occurrences)
.gb	.go
.gf	.go
.he	.hea
.if	.if
.in	.inl
.la	.la
.li	.bbl n
.ll	.pdw
.m1	.vmt
.m2	.vmh
.m3	.vmf
.m4	.vmb
.ma	.vmt/.vmb
.mp	.ps
.ms	.ls
.na	.all
.ne	.brn
.nf	.fif
.of	.foo
.oh	.heo
.op	.brp o
.pa	.brp n|+n
.pi	.bbp n
.pl	.pdd
.rd	.rd
.ro	.srm ro

.sk	.brs
.sp	.spb
.sr	.srv
.ss	.ls 1
.tr	.tr
.ts	.ts
.ty	.ty
.un	.unl
.ur	.ur
.wt	.wt

Attached to this MTB is a draft of a new MPM description of the runoff command.

NB Needless to say, this document and its supporting attachment were produced by the proposed implementation.

runoff

Name: runoff, rf

The runoff command is used to prepare formatted documents from text segments for production on various documentation devices including line printers and user terminals. Output pages are composed from various text blocks and controls provided in input files. Detailed control over page composition is provided by controls in the input file.

Details of the "source language" and various runoff controls are discussed later in this description.

Usage

runoff paths {-control_args}

where:

1. paths
   are the pathnames of input files named <entryname>.runoff. The runoff suffix must be the last component of the input file names; 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. Up to one hundred (100) input files may be processed with one invocation of the command. Output is produced in the order in which the pathnames are given in the command line. Input files are restricted to being single-segment files (SSF); output files for very large documents are converted to multi-segment files (MSF).

2. control_args
   may be chosen from the following list. Any control argument specified in the command line applies to all input file pathnames given. Control arguments may be freely intermixed with input file pathnames.

   -ball N, -bl N
   specifies the typeball in use on 2741-type terminals. If the user's login data does not indicate a 2741-like terminal type, the control argument is ignored. This control argument is identical to the "-device 2741-N" control argument except that "-device 2741-N" does not check the login terminal type.

   -character, -ch
   flags certain key characters and lines in the output by writing the composed output line to a segment named <entryname>.chars. Normal output is not affected. Page and line numbers referring to the normal output appear with each flagged line in <entryname>.chars. The set of key characters is controlled with the characters control in the input file. The default for this feature is OFF.

-device name, -dv name
prepares output compatible with the device specified. This control argument is used when the target device for output is not the default device for the output mode selected. The default value for name is "ascii". If the -file/-segment control argument is given, the default device is taken to be the online printer. If the -file/-segment control argument is not given, the default device is taken to be the user's terminal which is determined from the login data. Any device for which name.rf_device_table exists is a supported device.

-from N, -fm N
starts printed output at page N. If the -page control argument is given, printed output starts at the renumbered page N. The default value of N is 1.

-galley, -gl, -nopagination, -npgn, -no_pagination
produces galley format (continuous text without running headers and footers) without pagination. The default for this feature is OFF.

-hyphenate N, -hyph N, -hph N
changes the default hyphenation mode from OFF to ON. The optional parameter N is the length of the smallest separated word part. Its default value is 3.

-indent N, -in N
adds N spaces at the left margin of the output. This extra space has a default value of 20 if the output device is the online printer and has a default value of 0 for all other devices. The space given overrides the default value and is in addition to any indentation given with indent-left controls in the text.

-lines N1,N2, -li N1,N2
produces galley format output with line numbers only for input line N1 through N2. The default value of N1 is 1 and the default value for N2 is the last line in the input file. If N2 is not given, a comma need not be given. If N1 is not given, a comma must precede a given value for N2. The default for this feature is OFF.

-linespace N, -ls N
changes the default line spacing value to N. The line-space control uses N as a minimum value. The default value for N is 1.

-name string, -nm string string is the name of an input file even though it may have the appearance of a numeric parameter or a control argument.

-noart, -no_art
disables the conversion of conventional artwork constructs so that the details of those constructs may be seen in the formatted pages. The default for the artwork conversion feature is ON.

-nocontrol, -noc, -no_control
ignores all controls in the input files. All control lines are ignored and all embedded controls are removed from the text. Only raw text lines are composed as output. The default for this feature is OFF.

-nofill, -nof, -no_fill
ignores all fill and align controls in the input files. Only ragged text without regard to line length is composed. The default for this feature is OFF.

-number, -nb
prints input line numbers at the left margin of the output. The default for this feature is OFF.

-page N, -pg N
changes the initial page number to N. All subsequent pages are similarly renumbered. If a break-page control in the text gives a page number, the -page control argument is overridden and pages are numbered according to the break-page control. The default value for N is 1.

-parameter arg, -pm arg
assigns the string value arg to the internal variable "Parameter". The default value for arg is an empty string.

-pass N
processes the input file N times to permit proper evaluation of expressions containing variables that are defined following their reference(s) in the text. No output is produced until the last pass. The default value for N is 1.

-segment, -sm, -file, -fi
directs output to the file <entryname>.runout. This control 2 argument assumes by default that the document will be printed on the online printer (see the -device control argument above). An extra left margin space of 20 is assumed unless the -indent control argument specifies a different value or the -device control argument specifies a device other than the default device. The default for this feature is OFF.

-stop, -sp
waits for a newline character (ASCII NL) from the user before beginning the first page of output to the terminal and after each page of output including the last page. Any other characters typed are ignored, thus any forms positioning and top-of-form notes for special forms are easily accomplished. The default for this feature is OFF.

-to N
ends output after the page numbered (or renumbered) N. The default value for N is the last page.

-wait, -wt
waits for a newline character (ASCII NL) before beginning the first page of output to the terminal, but not between pages (see the -stop control argument above). The default for this feature is OFF.

Notes

1. A runoff input file contains intermixed text and controls. Controls are distinguished from text by their format; ".XXM <variable_field>". See "Preparation of Input Files for runoff" later in this description.

2. Summary of text controls. See "Preparation of Input Files for runoff" later in this description. (ø is a literal ASCII blank.)

.* {<string>}	comment
.~ {<string>}	comment
.al	align
ø	both (left/right)
b	both (left/right)
c	center
i	inside
l	left
o	outside
r	right
.bb	block-begin
ø	inline
a {<#>}	artwork
c {<#>}	centered
e {<#>}	equations
f {s}	footnote
i	inline
k {<#>}	keep
l {<#>}	literal
n {<name>}	named
p {<#>}	picture
.be	block-end
ø	all
a	art
c	centered
e	equations
f	footnote
k	keep
l	literal
n	named
p	picture
.bi <name>	block-insert
.br	break
ø	format
b	block
f	format
n {n}	need
p {e|o|n|+n}	page
s {n}{<string>}	skip
.cc c	change-character
.ch cd..	characters
.ex <expr>	execute
.fb	footer-block
ø {e|o|a}	begin
b {e|o|a}	begin
e	end
.fi	fill
ø	on
f	off
n	on
.fo	footer
ø {<#>}{±n}{title>}	all
a {<#>}{±n}{title>}	all
e {<#>}{±n}{title>}	even
o {<#>}{±n}{title>}	odd
.ft	footnote
h	hold
i	insert
r	running
u	unreferenced

runoff

```
.go <name>                                  go-to
.hb                                         header-block
   ø {e|o|a}                                   begin
   b {e|o|a}                                   begin
   e                                           end
.he                                         header
   ø {<#>}{±n}{title>}                         all
   a {<#>}{±n}{title>}                         all
   e {<#>}{±n}{title>}                         even
   f {<#>}{±n}{title>}                         footnotes
   o {<#>}{±n}{title>}                         odd
.ht                                         horizontal-tabs
   d {<name>} {ns,ns,ns,...}                   define
   f {ccc...}                                  off
   n {<name>} c                                on
.hy                                         hyphenate
   ø                                           default (from -hyphenate)
   f                                           off
   n                                           on
.if <name> {<expr>}                         insert-file
.in                                         indent
   l {±n}                                      left
   r {±n}                                      right
.la <name>                                  label
.ls {±n}                                    line-space
.pd                                         page-define
   ø {d,w}                                     all
   d {±n}                                      depth
   w {±n}                                      width
.ps {±n}                                    page-space
.rd                                         read
.rt                                         return
.sp                                         space
   ø {±n}                                      block
   b {±n}                                      block
   f {±n}                                      format
.sr                                         set-reference
   ø <name> <expr>                             variable
   c <name> <expr1> {by ±<expr2>}              counter
   m mode {<name>,<name>,...}                  mode
   v <name> <expr>                             variable
.tb                                         title-block
   b {c|h}                                     begin
   e                                           end
.tr cd..                                    translate
.ts {<expr>}                                test
.tt                                         text-title
   c {<#>}{±n}{title>}                         caption
   h {<#>}{±n}{title>}                         heading
.ty {<expr>}                                type
.un                                         undent
   ø {±n}                                      left
   l {±n}                                      left
   n {±n}                                      left-nobreak
   r {±n}                                      right
.ur <expr>                                  use-reference
.vm                                         vertical-margin
   ø {b,f,h,t}                                 all
   b {±n}                                      bottom
   f {±n}                                      footer
   h {±n}                                      header
   t {±n}                                      top
.wi                                         widow
```

ø {±n}	text
f {±n}	footnote
t	text
.wt	wait

3. Summary of builtin variables. See "Preparation of Input Files for runoff" later in this description.

Ad, AlignBoth	align-both mode flag
AlignCenter	align-center mode flag
AlignInside	align-inside mode flag
AlignLeft	align-left mode flag
AlignOutside	align-outside mode flag
AlignRight	align-right mode flag
Art	artwork block flag
BlockName	the name of the current text block
CallingFileName	entryname of calling file
Ce	count of lines to be centered
CharsTable	translation table for .chars file
Charsw	.chars mode flag
Date	current date
Device	device name
Eq	count of equation lines
Eqcnt	equation reference count
ExtraMargin	extra left margin value
Fi	fill mode flag
FileName	entry name of current input file
Filesw	file/segment output mode flag
Foot	footnote counter
FootRef	footnote reference string
Fp	number of first printed page
Fr	footnote reset mode
From	value of -from parameter
Ft	footnote mode flag
Galley	galley mode flag
Hyphenating	hyphentation mode flag
In	value of -indent parameter
IndentRight	value of right margin indentation
InputFileName	entry name of file being processed
InputLines	line number of current input file
Keep	keep mode flag
LinesLeft	text lines left on page
Ll	current line length
Lp	number of last printed page
Ma1	page top margin
Ma2	header margin
Ma3	footer margin
Ma4	page bottom margin
Ms	line spacing value
MultiplePagecount	count of form feeds between pages
NestingDepth	current insert file depth
Nl	line number on current page
NNp	next page number
NoFtNo	footnote number supression flag
NoPaging	no-pagination flag
Np	current page number
PadLeft	left/right pad switch (always 0)
Page	-page value
Parameter	passed parameter value
ParamPresent	passed parameter flag
Passes	-pass value
Pi	picture space waiting
Pl	current pagelength

Print	print flag
Printersw	-file/-segment control flag
PrintLineNumbers	line number control flag
Roman	roman page numbers flag
Selsw	2741 ball number
SpecCh	special delimiter character
Start	starting putput page number
Stopsw	-stop control flag
TextRef	footnote reference string
Time	command invocation time
To	-to value
TrTable	-chars translation table
Un	undent value
UndentRight	undent right margin value
UserInput	label value for builtin function
VMargTop	page top margin
VMargHeader	header margin
VMargFooter	footer margin
VMargBottom	page bottom margin
Waitsw	-wait control flag
Widow	current text widow size
WidowFoot	current footnote widow size

Preparation of Input Files for runoff

This section discusses the runoff control "language" and the preparation of runoff input files.

Output pages are composed from an optional header block, various text blocks, and an optional footer block. If all blocks for a page are empty (zero line count), a page number is skipped and no output is produced.

Output text blocks are constructed from input text and control strings; may consist of plain language paragraphs, lineart diagrams, ruled tables, equations, optional footnotes, or space reserved for hand-art or picture addition; may be left and/or right justified or centered; and may be placed arbitrarily on the page. There are two types of text blocks; the inline block that is composed into the output immediately upon the occurrence of a block break, and the named block that is held for later insertion.

Header and footer blocks consist of top and bottom page margin space and up to 20 lines of text. They may be specified the same or separately for odd and even pages, and each text line may contain a left margin part, a centered part, and a right margin part.

Footnote blocks consist of collections of specially designated text lines that are placed between the last main body text block on a page and the optional footer block and may be composed with a format different from that used for the main body text. Footnotes may be printed page-by-page as they occur or may be held for insertion as the user chooses. The default is page-by-page. Any pending held footnotes are inserted at the end of the document.

Pages may be numbered from any arbitary starting page number and page numbers may be printed in any of the numeric display modes (see set-reference-mode control below).

Detailed control of page composition is provided by controls in the input file. Controls have the form ".XXM <variable_field>" and may occupy an input line by themselves or be embedded in the text as a delimited reference string as appropriate for the particular control. Output may be directed back to the user's terminal or to a file for eventual transcription to another medium (online printer or magnetic tape, for example). If the output is directed back to the user's terminal, it may be printed page by page to allow positioning of special forms. Terminal devices with the full 95 character ASCII graphics set are fully supported. For other devices having a limited character set, there is a facility for replacing any character (or set of characters) with blanks or any other characters of the user's choice. If special symbols are to be hand-drawn, a separate segment with hand-art instructions can be created. The user can define variables and cause expressions to be evaluated. The user also has the ability to refer to (and sometimes modify) variables connected with the functioning of the program.

INPUT FILE ORGANIZATION

A runoff input file contains intermixed text and controls. Controls are distinguished from text by their format; ".XXM <variable_field>". XXM is chosen from the set of control mnemonic codes given below and <variable_field> depends on the requirements of the particular control. One or more blanks must separate XXM and <variable_field>.

If an input file line has a period in column one, the line is a control line and may contain one and only one control. Controls may be embedded in the text by enclosing them between special delimiter characters (see "Special Delimiter" and "Embedded Controls" later in this description). Embedded controls have that same effect as control lines except for possible space insertion due to an end-of-line condition.

Input file lines starting with any character other than a period are processed as text lines. If an input file text line is too short or too long to fill an output line, text material is taken from or deferred to the next output line (unless the fill mode is specifically disabled). A line starting with white space (ASCII SP or HT character) causes a format break. An empty line or a line containing only white space (one or more ASCII SP or HT characters) causes a block break and generates a blank line in the output.

Tab characters (ASCII HT) encountered in the input file lines are replaced with that number of blanks required to reach the next Multics standard tab column (11, 21, 31, ...). (See the horizontal-tabs control below for a discussion of tabulation in the output.)

Normally, when in fill mode and a format break has not occurred, each new-line (ASCII NL) character in the input file is replaced with a single blank (ASCII SP). When an input text line ends with any of the characters ".", "?", "!", ";", or ":", or with ".", "?", or "!" followed by a double quotation mark 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 allowed on any input or output line is 1800. This value easily accommodates a twelve inch line at the sixty-per-inch pitch of various plotting terminals.

TERMINOLOGY

The following paragraphs describe various terms used or implied throughout the runoff description.

Text Blocks

A text block is a block of composed output that is treated as an entity. It is formed by accumulating text from the input file until an explicit or implicit block break is encountered. The text block is the basis for widow processing. The widow size specifies the minimum number of text block lines that may be split away from the block for distribution. No text block containing less than twice the widow size is ever split. Text blocks containing at least twice the widow size are broken in such a way that each part contains at least the widow size. The default widow size is two lines for main body text and one line for footnotes. The widow size may be changed by the user with the widow control.

Under certain conditions, the processing of a text block may be "suspended". When a block is suspended, certain items associated with the block are set aside (or "pushed") for possible resumption of processing in that block. If the block is resumed, those items are restored to an active state (or "popped") and processing of the block continues. The items involved are the text alignment mode, the fill mode, the line length, the line spacing value, and the block mode (keep, picture, equation, etc.).

Breaks

A break is an event that causes an interruption of some processing mode. Three different breaks are defined.

Format break
: This break is caused by a text control that changes or interrupts the current formatting mode, but does not define a text block. Examples are indent, undent, page-define-width, and break-format. Any pending input text is composed into the output as a short line. The current text block is continued with the new formatting mode.

Block break
: This break is caused by a text control that defines a text block. Examples are space-block, break-block, and break-page. The current text block is terminated (as appropriate) with a format break, written out, and a new text block of the type specified is begun.

Page break
: This break is caused by a text control that forces a new page. Examples are break-page and break-need. A page break ensures that no text following the control causing the break is printed on the current page. If inline text is being processed, the current page is closed out (with footnotes and footers as appropriate). Any pending text is handled according to the control given (see control descriptions below).

Fill and Align Modes

The actions of fill mode and align-both mode are interrelated. In fill mode, text is moved from line to line when the input text line either exceeds or cannot fill an output line. In align-both mode, uniformly distributed extra space is inserted into the filled lines until the text is even at both margins. Initial white space on a line is not subject to alignment. For an undent control, the characters moved to the left of the established left indentation point are not subject to alignment. Unfilled lines (including any short lines at the ends of text blocks) are not aligned. Align-both mode is not enabled unless fill mode is ON although fill mode be enabled without align-both mode, yielding filled and ragged-right, ragged-left, or ragged-both output text depending on the the modifier of the align control currently in effect.

Line Length

The line length is the space available for text in an output line, including all spaces and indentations, but not including margins set or implied by the -device, -indent, or -number control arguments. Space is measured in units of 10-pitch characters (10 characters = 1 inch).

Spacing between Lines

Vertical spacing within a text block is controlled by the line-space control. A line-space control with a value of N inserts N-1 line spaces between text lines.

Vertical Margins

There are four vertical margins on a page. Their descriptions, controls, and default values are:

Margin	Control	Default value
Between top-of-page and first header	.vmt	4
Between last header and first text line	.vmh	2
Between last text line and last footer	.vmf	2
Between first footer and bottom-of-page	.vmb	4

The actual space appearing at the top- and bottom-of-page margins may vary among devices because of differing physical constraints.

Page Numbers

As the output is being prepared, a page number counter is kept. This counter can be modified by the user with the break-page control. The current value of the counter can be referenced by the use of a single, unpaired special delimiter character in the several header and footer controls and in use-reference controls. A page is called odd or front if the value of the page counter is odd and even or back if the value is even. Page numbers can be printed in any of the numeric display modes (see set-reference-mode control below). The default is Arabic.

Headers and Footers

A header line is a line printed at the top of each page. A footer line is a line printed at the bottom of each page. A page may have a header block and/or a footer block, each containing up to twenty lines. Header lines are numbered from the top down, footer lines from the bottom up. The two blocks are completely independent of each other. Provision is made for different headers and footers for odd and even pages. Both odd and even header (footer) lines are set by use of the header (footer) control. They may be set separately by use of the even and odd modifiers of the header and footer controls.

The <variable_field> of a header or footer control may contain three optional parameters; a line number, a left margin adjustment, and a title. They must be separated by one or more blanks.

The line number specifies which header or footer line is being set. If the line number is omitted or has the value 0, all previously defined headers or footers of the type specified (odd, even, or both) are cancelled. If the line number is given it must have a value less than or equal to to 20, inclusive, and only the line specified is affected by the control. Once set, a line is printed on each output page until it is reset or cancelled.

The left margin adjustment must have the form ±N. If it is given without the sign, the value is the column at which to align the title. If it is given with the optional sign, the value is a local adjustment to the current left indentation value. If it is omitted, the title is aligned at the current left indentation point. A left margin adjustment may not be given unless a line number is also given.

The title begins at the first non-blank, non-numeric (including signs) character in the variable field. This character is used as a local delimiter (effective only for the line being processed) and may be any non-numeric character not used in the text of the title. The local delimiters divide the title text into three parts, a left margin part, a centered part, and a right margin part, which must be given in that order. Local delimiters for parts lying to the right of the last desired part may be omitted. If two successive local delimiters are given, the corresponding title part is set to an empty (zero length) string. If the title consists of one or more occurrences of a single character, then all parts are empty and the affected line is reset to a blank line. If the title is omitted, the affected line is reset to a null line. Null lines are not printed.

If both the line number and title are omitted, the entire header or footer block of the type specified (even, odd, or both) is set to empty (no printed lines).

An occurrence of a single, unpaired special delimiter character in a title part for a header or footer line is replaced with the current value of the page counter when the line is printed. If multiple page number references or the values of other variables are desired in a title part, the value of the program variable "Np" must be substituted for the page number reference(s) (see "Program Variables" later in this description).

Text Titles

Text titles are sequences of title lines that provide a heading or caption for a figure, ruled table, section, or paragraph in a document. For the purposes of composition, they are treated as an integral part of the text block to which they apply. There are four controls for the specification of text titles.

The number of heading lines is added to the widow size at the beginning of a text block and the number of caption lines is added to the widow size at the end of a text block for widow processing. Thus, if a text block must be moved or split to prevent widows, the associated heading and/or caption is moved with the parts of the split text block, that is, the title is never separated from the text block.

Special Delimiter

One character of the 95 character ASCII set is designated as a special delimiter. The default character is percent ("%") and may be changed at any time by the user with the change-character control. Special delimiters are used to enclose a variable name to form a symbolic reference to the value of the variable. These symbolic references are replaced with the corresponding values during variable substitution. Single, unpaired occurrences of the special delimiter character in a control or title part signify a reference to the value of the page counter.

Hyphenation

The algorithm for word hyphenation is based on a dictionary search. The user has control over hyphenation (down to the line-by-line level) with the hyphenate-on and hyphenate-off controls. The -hyphenation control argument changes the default hyphenation mode from OFF to ON and allows specification of the smallest separated word part.

EXPRESSIONS AND EXPRESSION EVALUATION

An expression can be numeric, string, or relational and consists of symbolic variable references, literal numbers, literal strings, and operators in

appropriate combinations. All numeric operations are performed in fixed point decimal mode with precision (11,2).

The defined operators are (in order of precedence):

^ (Boolean NOT), & (Boolean AND), ¦ (Boolean OR), ≡ (Boolean EXCLUSIVE OR)

- (unary negation), * (multiplication), / (division), \ (remaindering)

+ (addition), - (subtraction)

= (equal), < (less than), > (greater than)

≠ (not equal), ≤ (less than or equal), ≥ (greater than or equal)

Arithmetic operations yield fixed point decimal results, relational operations yield logical true or false results, and string operations yield string results of the length of their longest operand.

Other guidelines for the use of expressions are:

1. Parentheses may be used for grouping.
2. Blanks outside of literal strings are ignored.
3. Octal integers consist of "#" followed by a sequence of octal digits.
4. Literal strings are enclosed by double quote characters. Certain characters whose literal occurrence is wanted in the string must be given with a conventional escape sequence beginning with the "*" or "^" characters as follows:

**	asterisk character
*^	caret character
*"	double quote character
*b	backspace character (ASCII BS)
*n	newline character (ASCII NL)
*s	blank character (ASCII SPACE)
*t	horizontal tab character (ASCII HT)
*f	formfeed character (ASCII FF)
*c*nnn*	the ASCII character whose decimal value is *nnn* (1 to 3 digits) (may also be given as *c#*nnn* with *nnn* in octal)

   The "*" characters are removed during escape processing while the "^" characters are retained until the string is inserted into an output line. This feature provides a "reconceal" function.
5. Concatenation of strings is performed from left to right.
6. For <string_expression> of length *l* and positive *i* and *k*;

<string_expression>(i) is a string of length (l-i+1) beginning with the ith character of <string_expression>, and

<string_expresssion>(i,k) is a string of length k beginning with the ith character of <string_expression>.

If i is negative, the string begins at the -ith character before the end of <string_expression> and the length is set accordingly. If k is negative, the string ends at the -kth character before the end of <string_expression> and the length is set accordingly. In all cases, the kth character must lie to right of the ith character in <string_expression>.

7. Evaluation of substrings as defined above takes place after any concatenations; arithmetic operations have higher precedence than all relational operations, and string operations have higher precedence than all the arithmetic operations.

8. If a string value appears where a numeric value is required, or vice versa, conversion is attempted to the mode required by the operator. If the conversion is unsuccessful, an error diagnostic message is produced.

9. Expression evaluation takes place after variable substitution for those controls allowing substituion of variables (see control descriptions following).

DEFINITION AND SUBSTITUTION OF VARIABLES

User variables can be defined with the set-reference-variable and set-reference-counter controls and their values retrieved thereafter by a symbolic reference. The names of variables are constructed with the alphanumeric characters, decimal digits, and "_" with a maximum length of 32 characters. When a variable is defined, it is given a type (string or numeric) based on the control and the type of the expression that is to be its value. Variables that are undefined at the time of reference yield an empty string or a numeric zero depending on the required type.

In substitution of variables, the name of the variable is enclosed by special delimiter characters. A single, unpaired occurrence of the special delimiter character is replaced by the current value of the page counter. If the literal occurrence of the special delimiter character is wanted in a line that is subject to substitution, it must be be given as a unpaired, doubled character ("%%" for the default character).

Substitution of variables takes place:

1. In all controls for which automatic substitution is specified (see control definitions in "Controls" below).

2. In all header, footer, equation, and text title lines.

Many of the variables internal to runoff are available to the user (a complete list is given later in this description). These variables include control argument values (or their defaults), values of switches and counters, and certain tables. However, the user need not be concerned about naming conflicts since an attempt to modify the value of an internal variable by a means other than the control provided for such modification is detected and causes an error

diagnostic message. Such errors are non-fatal but may produce anomalous results.

Two special builtin counters are provided for use in footnote and equation numbering. "Foot" contains the value of the next footnote number available (or the current footnote number if the reference is from within the text of the footnote) and "Eqcnt" contains the value of the next available equation number. The value of "Foot" is incremented when the closing block-end or block-end-footnote control for a footnote is encountered. The footnote counter may be allowed to run continuously for the entire document or may be reset whereever the user desires (see the footnote control below). Any reference to "Eqcnt" returns the current value and increments the counter. Therefore its value should be assigned to a variable and that variable be used in referring to the equation.

User defined counters are controlled with the set-reference-counter control in which an initial setting and an incremental value are specified. Each reference to a user defined counter returns its current value and changes the value by the specifed increment.

Three special cases of variable reference are defined.

1. If the name of the variable is "UserInput", the value is the string read from the stream user_input. This feature allows direct, interactive control of the text formatting process. If the user types in a control, that control will be processed just as though it had been read from the input file.

2. If the name of the variable is "[active_function_name]", the value is the string returned by the active function (see Section II, "Active Functions," of MPM Commands).

3. If the name of the variable begins with a period, the delimited string is taken to be an embedded control with the defined control format instead of a variable reference and is passed to the text control processor.

EMBEDDED CONTROLS

Under certain conditions it may be necessary or desirable to embed a control within the text without the space implied by the end of an input text line. A prominent case is the shift to italics or boldface within a word for emphasis. The capability is supported by permitting controls to appear within text as pseudo-variables. The construct is:

.ur <text>%.XXM <variable field>%<text> ...

When this construct is encountered, the embedded control is processed just as though it had occurred in a normal control line.

DEFAULT CONDITIONS

When no control arguments are given, runoff sets all internal variables and control parameter values to the default values shown in their respective descriptions. (See "Controls" following and "Usage" earlier in this description.)

The control arguments establish a modified set of default values for the invocation of the command. The working values of the internal variables may be further modified by the text controls. If multiple input files are given, all values are reset to the modified default values for each input file.

CONTROLS

This section gives descriptions of the controls available in runoff. Each description consists of a general control explanation followed by explanations of the various modified forms.

Every explanation has a title line giving the control or modifier name, the mnemonic code and possible variable field template, the break type generated (if any), and the substitution of variables mode. Modifier explanations are indented to show their subordination to the control.

The template for the <variable_field> may contain the following symbolized parameters:

<#>	an integer constant
<±n>	a numeric expression with an optional leading sign
<expr>	an arbitrary expression (string, logical, or numeric)
<c>	any single character
<cd>	any character pair
<name>	a name string up to 32 alphanumeric characters
<string>	an arbitrary character string up to 1800 characters
<title>	a three part title line of the form *part1*part1*part3* where "*" may be any nonnumeric character not appearing in the parts

If any parameter appears without the enclosing less-than, greater-than (<>) signs, it is a literal and must appear as shown. Vertical strokes (|) separate the choices, if any, for literal parameters. Parameters shown within braces ({}) are optional. An elipsis (...) indicates continuation of a parameter string to the entent given in the explanation.

align: .al{m}

Align the text within the defined text area on the page according to the modifier given. Text processing is interrupted with a format break, any pending text is processed as a short line, then processing is resumed on a new line in the same text block with the new alignment mode. When any form of the control is given, the mode for that form is set ON and all other alignment modes are set OFF. The fill mode may be either ON or OFF. If the fill mode is OFF, overlength lines are not truncated. The default alignment mode is align-both (.alb).

both: .al, .alb; no parameters, format break, no substitution

Align the text at both the left and right margins according to the current values of left and right indentation and undentation. Text is padded by insertion of uniformly distributed white space. The fill mode must be ON for this mode to operate. If the fill mode is OFF, this mode is identical to the align-left (.all) mode. This is the default alignment mode.

center: .alc; no parameters, format break, no substitution

Center the text in the space defined by the current values of left and right indentation and undentation leaving both the left and right margins ragged.

inside: .ali; no parameters, format break, no substitution

Align the text on the inside margin (binding edge) according to the current values of the appropriate indentation and undentation leaving the outside margin ragged.

left: .all; no parameters, format break, no substitution

Align the text on the left margin according the current values of left indentation and undentation leaving the right margin ragged.

outside: .alo; no parameters, format break, no substitution

Align the text on the outside margin (away from binding edge) according to the current values of the appropriate indentation and undentation leaving the inside margin ragged.

right: .alr; no parameters, format break, no substitution

Align the text on the right margin according to the current values of right indentation and undentation leaving the left margin ragged.

block-begin: .bb{m}

Interrupt processing of the current block and either begin processing a new block or continue the current block according to the modifier given.

art: .bba; {<#>}, no break, no substitution

Begin flagging output text lines as artwork lines to be processed by the artwork expander function. If # is given, then flag exactly # output lines. If # is not given, then continue flagging until the occurrence a block-end or block-end-artwork control. This control form is disabled if the -noartwork control argument is given. If the artwork feature is disabled, no lines are flagged and the block is treated as a normal text block.

centered: .bbc; {<#>}, format break, no substitution

Cause a format break, processing any pending text as a short line in the current alignment mode. If # is given then suspend the current alignment mode and fill modes and accumulate exactly # unfilled output text lines aligned on the center column of the available text area on the page. Control lines are not counted, regardless of whether they add lines to the output or not. Fur-

ther, establish the # output lines so accumulated as a keep block within the current text block. If # is not given, then change the current alignment mode to be aligned on the center column of the available text area on the page and continue until a control changes the alignment mode.

equation: .bbe; {<#>}, format break, no substitution

Cause a format break, processing any pending text as a short line in the current alignment mode, then begin processing input lines as equation lines in the current text block. Equation lines must have the <title> format as discussed previously. The equation line parts will be aligned at the left and right indentation points. If # is given then suspend the current block mode and accumulate exactly # output equation lines. Control lines are not counted, regardless of whether they add lines to the output or not. Further, establish the # equation lines so accumulated as a keep block within the current text block. If # is not given, then change the current block mode to equation mode and continue until the occurrence of a block-end or block-end-equation control.

footnote: .bbf; {s}, no break, no substitution

Suspend processing of the current text block and begin processing a footnote. The text processing mode parameters are carried forward from the previous footnote or from the footnote defaults if no previous footnote has occurred. Any modes set while processing footnotes carry forward to all subsequent footnotes. If the literal parameter "s" (for "suppress") is given, the footnote reference (e.g. "(2)") is omitted and the footnote counter is not incremented.

inline: .bb, .bbi; no parameters, block break if inline, no substitution

If the current text block is an inline block, then cause a block break and begin a new inline block. If the current text block is a named block, then suspend copying that block and revert to inline block processing.

keep: .bbk; {<#>}, format break, no substitution

Cause a format break, processing any pending text as a short line in the current alignment mode, then establish subsequent output lines as an unbreakable keep block within the current text block. Keep blocks not subject to being split between pages. If # is given, then accumulate exactly # lines into the keep block regardless of whether they are caused by text or controls. If # is not given, then continue keep block accumulation until the occurrence of a block-end or block-end-keep control. The break-page and break-need controls are disabled while processing in this mode. If the number of accumulated output lines exceeds the maximum text space available on a page as determined by the vertical margins and any headers and footers, an error diagnostic message is produced and the block is broken into full and partial pages.

literal: .bbl; {<#>}, no break, no substitution

Begin processing input lines as text lines in the current text block even though they may have control format. If # is given then process exactly # input lines. If # is not given, then

continue literal line processing until the occurrence of a block-end or block-end-literal control.

named: .bbn; {<name>}{,a|r}, no break, no substitution

Begin copying input lines into the temporary insert file <name> and hold them for later inline insertion with the block-insert control as described below. The optional suffix is not part of the name but indicates what action is to be taken on the new block. "a" specifies that new lines are to be appended to any existing lines; "r" specifies that new lines are to replace the existing lines. The default action is append. The copying of input lines continues until the occurrence of any form of a block-end or block-begin control that changes to some other block, either named or inline. If <name> is not given, this form is identical to the block-begin-inline (.bbi) form.

picture: .bbp; {<#>}, no break, no substitution

If # is given, then define an unbreakable picture block of exactly # lines of white space. If # is not given, then accumulate output lines into an unbreakable picture block until the occurrence of a block-end or block-end-picture control. Text headings and/or captions given while in picture mode (# not given) pertain to the picture and not to a possible containing text block. A picture block is a white space or formatted block that is inserted on a space available basis. If, at the completion of a picture block, sufficient space remains on the current page, it is inserted immediately, including into the middle of the current text block. If the picture block does not fit on the current page, inline text is "promoted" ahead of the picture and the picture is inserted at the top of the next page. If the size of a picture block exceeds the maximum text space available on a page as determined by the vertical margins and any headers and footers, an error diagnostic message is produced and the block is broken into full and partial pages. Multiple picture blocks are queued, not merged into a single block. Queued picture blocks are inserted in the order in which they were defined.

block-end: .be{m}

Stop processing text into the current text block and/or in the current mode as determined by the modifier given.

all: .be; no parameters, block break if inline, no substitution

Suspend and/or stop processing in all of the modes discussed for the block-begin- control above, saving the current formatting parameters if appropriate, and revert to inline block processing. If none of the block modes are in effect, then cause a block break and begin a new inline block. This form of the block-end-control is the "broadside" form allowing the user to "back out" of an arbitrarily nested set of blocks without having to be concerned about the order in which they were begun.

art: .bea; no parameters, no break, no substitution

Stop flagging output lines for artwork conversion. If the artwork mode is not in effect, the control is ignored.

centered: .bec; no parameters, format break, no substitution

Stop processing the centered block as begun by a previous block-begun-centered control and revert to normal text processing. If equation mode is not in effect, then ignore the control.

equation: .bee; no parameters, format break, no substitution

Stop processing equation lines and revert to normal text processing. If equation mode is not in effect, then ignore the control.

footnote: .bef; no parameters, no break, no substitution

Stop processing the current footnote unit and revert to main body text, saving the footnote text processing mode parameters and restoring the main body text processing mode parameters. Increment the footnote reference counter unless the footnote- control or the "s" optional parameter on the block-begin-footnote control have specified that no reference to this footnote is to be made. If not in footnote mode, then ignore the control.

keep: .bek; no parameters, no break, no substitution

Stop counting output lines for the unbreakable keep block begun with the block-begin-keep control and reactivate the break-page and break-need controls. If the keep mode is not in effect then ignore the control.

literal: .bel; no parameters, no break, no substitution

Stop processing all input lines as text lines regardless of format and revert to normal control processing. This is the only control recognized while in literal mode. If the literal mode is not in effect then ignore the control.

named: .ben; no parameters, no break, no substitution

Stop copying input lines into the current named block and resume inline block processing. If the current text block is not a named block, then ignore the control.

picture: .bep; no parameters, no break, no substitution

Stop accumulating output lines into the current picture block and revert to inline block processing. If the picture will fit in the space remaining on the current page, then insert it immediately; otherwise, queue the picture block for insertion on a space available basis. If not in picture mode, then ignore the control.

block-insert: .bi; <name>, no break, no substitution

Suspend processing of the current input file and process the named block <name>. If <name> is not given or <name> does exist, an error diagnostic message is produced. The named block <name> is treated as though it were an external insert file (see the insert-file control below).

break: .br{m}

Interrupt processing according to the modifier given, then resume processing with the same processing modes.

block: .brb; no parameters, block break, no substitution

Terminate the current text block and insert it into the output document subject to the current processing modes. Any pending text is formatted as a short line. This control has no effect when processsing other than inline blocks.

format: .br, .brf; no parameters, format break, no substitution

Interrupt text processing, then resume with a new line in the current text block with the current modes. Any pending text is formatted as a short line.

need: .brn; {<#>}, possible block break, no substitution

If the number of available text lines left on the page is less than # then cause a page break. Any pending text is not processed but is deferred to the new page. The default value for # is 1.

page: .brp; {e|o|<#>|<±n>}, page break, no substitution

Terminate the current text block and the current page, then begin a new page according to the parameter given. Any pending text is formatted as a short line in the current text block. If the parameter is "e", then set the page number for the new page to the next even value. If the parameter is "o", then set the page number for the new page to the next odd value. If the parameter is #, then set the page number for the new page to #. If the parameter is ±n, then the current page number is changed by n to obtain the new page number. If no parameter is given, then the page number for the new page is the next sequential page number. No separating blank pages are produced.

skip: .brs; {<#>}{<string>}, page break, no substitution

Terminate the current text block and the current page, then produce # sequentially numbered blank pages (with headers and footers as appropriate). The default value for # is 1. If <string> is given, it is printed as a centered text block on the blank pages.

change-character: .cc; c, no break, no substitution

Change the special delimiter character used to delimit variables for substitution and for reference to the page counter to "c". The special delimiter character previously defined (including the default special delimiter character) is treated as a normal character. If "c" is omitted, the special delimiter character reverts back to the default special delimiter character. The default special delimiter character is "%".

characters: .ch; <cd><cd>..., no break, no substitution

Flags the occurrence of "c" by replacing it with "d" and writing the output line to the <entryname>.chars file (see "Usage" earlier in this description). The normal composed output is not affected. An

unpaired "c" at the end of the <variable_field> is treated as though it were paired with a blank. If any "d" is a blank, the corresponding "c" appears as itself in the <entryname>.chars file. If all "c"s in an output line are paired with blanks, the line is not written to the <entryname>.chars file. The default set of character pairs is empty.

comment: .*; {<string>}, no break, no substitution

A comment line having no effect on any output.

comment: .~; {<string>}, no break, no substitution

A comment with no effect on normal output. <string> is written to the <entryname>.chars file if the -character control argument is given. If <string> is not given, a null line is written.

execute: .ex; <string>, no break, no substitution

<string> is passed to the Multics command processor for execution as a command line.

fill: .fi{m}

Set the fill mode ON or OFF according to the modifier given. In fill mode, text words are moved from line to line in such a way that the last word, or parital word if hyphenation mode is in effect (see the "hyphenate-on" and "hyphenate-off" controls below and the -hyphenate control argument under "Usage" eariler in this document), does not violate the right margin. The default for this mode is ON.

off: .fif; no parameters, format break, no substitution

Set the fill mode OFF.

on: .fi, .fin; no parameters, format break, no substitution

Set the fill mode ON.

footer: .fo{m}

Define page footer lines according to the modifier given. The following actions are taken on the parameters for each of the modifiers. If <title> is omitted, then line # is replaced with a null line and the original numbering of lines in the footer block is not changed. If <title> consists of one or more occurrences of the same character, then line # of the footer block is replaced with a blank line. If n is omitted, then <title> is aligned according the value of left indentation at the time of footer insertion. If n is given without the optional sign, then <title> is aligned at column n. If n is given with the optional sign, then n is used as a adjustment to the value of left indentation at the time of footer insertion. n may not be given unless # is also given. If # is omitted, then the footer block is cancelled and <title> becomes line 1 of a new footer block. If # is larger than the value of the next footer line, then intervening null lines are inserted. If both # and <title> are omitted, then the footer block is cancelled. Footer block lines are numbered from the bottom up. Default footer blocks are empty.

all: .fo, .foa; {<#>}{±n}{<title>}, no break, substitution when inserted

Define footer lines for all pages.

even: .foe; {<#>}{±n}{<title>}, no break, substitution when inserted

Define footer lines for even pages only.

odd: .foo; {<#>}{±n}{<title>}, no break, substitution when inserted

Define footer lines for odd pages only.

footer-block: .fb{m}

Define a footer block according to the modifier given.

begin: .fb, .fbb; {e|o|a}, no break, no substitution

Suspend text processing, then cancel the footer block of the type selected by the given parameter, and define a new footer block of the same type according to the input lines following. If the parameter is "e" then define a footer block for even pages; if the parameter is "o" then define a footer block for odd pages; if the parameter is "a" then define a footer block for all pages. Processing of lines into the footer block will continue until the occurrence of a footer-block-end control. Input lines for the footer block may be text lines, controls, or <title> lines. <title> lines may contain the optional left margin adjustment given before the actual <title> and substitutable variables. Substitution is done when the footer is inserted.

end: .fbe; no parameters, no break, no substitution

Stop processing a footer block as begun by a preceding footer-block- control and revert to text processing. If not in footer block mode, then ignore the control.

footnote: .ft{m}

Controls footnote positioning and numbering according to the modifier given.

hold: .fth; no parameters, no break, no substitution

Do not insert footnotes on the page of their reference, but hold them aside for insertion by the user with the footnote-insert control. The footnote counter runs continuously until reset by the footnote-insert control.

insert: .fti; {<#>}, no break, no substitution

Insert all pending footnotes and reset the footnote counter to #. The default value for # is 1.

running: .ftr; {<#>}, no break, no substitution

Insert footnotes as they appear and on the page of their reference. The footnote counter is reset to # at the top of each new page. The default value for # is 1.

unreferenced: .ftu; no parameters, no break, no substitution

Begin unreferenced footnote mode for all following footnotes. Footnotes are not numbered, the footnote counter is not incremented, and footnote references are not set into the text. Unreferenced footnotes are inserted according to the footnote-hold or footnote-running controls. Unreferenced footnote mode continues until the occurrence of a footnote-hold or footnote-running control.

go-to: .go; <name>, no break, no substitution

Reposition the input file to the line containing the label control having <name> as its <variable_field>. The label control with that <name> should be unique within the file for correct operation. If <name> is not unique, the file is positioned to the first ".la <name>" line. If <name> is not a label defined in the file, an error diagnostic message is produced, an end-of-file condition is simulated, the current file is closed, and text processing resumes either with the line following the insert-file control of the "calling" file or the next input file. If <name> is defined, text processing resumes with the input line following the ".la <name>" line.

header: .he{m}

Define page header lines according to the modifier given. The following actions are taken on the parameters for each of the modifiers. If <title> is omitted, then line # is replaced with a null line and the original numbering of lines in the header block is not changed. If <title> consists of one or more occurrences of the same character, then line # of the header block is replaced with a blank line. If n is omitted, then <title> is aligned according the value of left indentation at the time of header insertion. If n is given without the optional sign, then <title> is aligned at column n. If n is given with the optional sign, then n is used as a adjustment to the value of left indentation at the time of header insertion. n may not be given unless # is also given. If # is omitted, then the header block is cancelled and <title> becomes line 1 of a new header block. If # is larger than the value of the next header line, then intervening null lines are inserted. If both # and <title> are omitted, then the header block is cancelled. Header block lines are numbered from the top down. Default header blocks are empty.

all: .he, .hea; {<#>}{±n}{<title>}, no break, substitution when inserted

Define header lines for all pages.

even: .hee; {<#>}{±n}{<title>}, no break, substitution when inserted

Define header lines for even pages only.

footnote: .hef; {<#>}{±n}{<title>}, no break, substitution when inserted

Define the header for footnotes. If <title> is omitted, the footnote header line is reset to the default string value. The default footnote header line is a string of underscore characters from the left margin to the right margin.

odd: .heo; {<#>}{±n}{<title>}, no break, substitution when inserted

Define header lines for odd pages only.

header-block: .hb{m}

Define a header block according to the modifier selected.

begin: .hb, .hbb {e|o|a|f}, no break, no substitution

Suspend text processing, then cancel the header block of the type selected by the given parameter and define a new header block of the same type according to the input lines following. If the parameter is "e" then define a header block for even pages; if the parameter is "o" then define a header block for odd pages; if the parameter is "a" then define a header block for all pages; if the parameter is "f" then define a header block for footnotes. Processing of lines into the header block will continue until the occurrence of a header-block-end control. Input lines for the header block may be text lines, controls, or <title> lines. <title> lines may contain the optional left margin adjustment given before the actual <title> and substitutable variables. Substitution is done when the header is inserted.

end: .hbe; no parameters, no break, no substitution

Stop processing a header block as begun by a preceding header-block- control and revert to text processing. If not in header block mode, then ignore the control.

horizontal-tab: .ht{m}

Define and control horizontal tabulation according to the modifier given. Up to 20 horizontal tabulation stop patterns may be in effect at any one time.

define: .htd; <name> {<#><s>,<#><s>,<#><s>,...}, no break, no substitution

Define pattern <name>, setting tab stops at columns (<#>,<#>,<#>,...) inclusive. Up to 20 columns may be set for the pattern independently of any other pattern(s) set. Each <#> given may have associated with it a string, <s>, that specifies the character pattern to fill any space ahead of the tab stop column. <s> is repeated as necessary and is positioned so that the last character of <s> is in the column just before the tab stop column. <s> may quoted or unquoted. The default fill string is blank characters. If no tab stop columns is given, the pattern entry for <name> is cancelled. If no <variable_field> is given, all tab stop patterns are cancelled.

off: .htf; {<c><c>...}, format break, no substitution

Disable horizontal tabulation processing for the character(s) <c>. If no tab stop pattern has been enabled for a given <c>, then that character is ignored. If no <c>'s are given, then horizontal tabulation processing is disabled for all current patterns.

on: .htn; <name> <c>, format break, no substitution

Enable horizontal tabulation processing for pattern <name> and associate the character <c> with the pattern. If <name> is not given or is not a defined tabulation pattern, or if <c> is not given, then an error diagnostic message is produced.

hyphentate: .hy{m}

Control hyphenation according to the modifier given.

default: .hy; no parameters, no break, no substitution

Set the hyphenation mode to the default value (ON or OFF) as established by the -hyphenation control argument (see "Usage" earlier in this document).

off: .hyf; no parameters, no break, no substitution

Set the hyphenation mode OFF.

on: .hyn; no parameters, no break, no substitution

Set the hyphenation mode ON.

indent: .in{m}

Control the positioning of text with respect to the left and right margins according to the modifier given.

left: .in, .inl; {<±n>}, format break, no substitution

If n is given without the optional sign, then set the left indentation point to n columns to the right of the left margin. The left margin is determined by the physical characteristics of the output device and any possible value given or implied with the -indent or -device control arguments. If n is given with the optional sign, then change the current left indentation point by n columns. Positive values for n cause movement to the right. Any combination of parameters that results in a zero or negative effective line length will produce an error diagnostic message. The left indentation point will never be set to the left of the left margin. The default value for n is 0.

right: .inr; {<±n>}, format break, no substitution

If n is given without the optional sign, then set the right indentation point to n columns to the left of the right margin. The right margin is determined by the physical characteristics of the output device, any possible value given or implied with the -indent or -device control arguments, and the page-define-width control. If n is given with the optional sign, then change the current right indentation point by n columns. Positive values for n cause movement to the left. Any combination of parameters that results in a zero or negative effective line length will produce an error diagnostic message. The right indentation point will never be set to the right of the right margin. The default value for n is 0.

insert-file: .if; <name> {<expr>}, no break, no substitution

Processing of the current input file is suspended and file <name>.runoff is opened and processed starting with line 1 of that file. <expr> is evaluated and its value placed in the program variable "Parameter" for use by the inserted file (any existing value in "Parameter" is destroyed). When the processing of the inserted file is complete (see the return control below), processing of the sus-

pended input file is resumed with the line following the line containing the insert-file control. Files may be inserted to a nested depth of 50.

label: .la; <name>, no break, no substitution

Establish <name> as a target for possible go-to controls.

line-space: .ls; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the linefeed count to n. If n is given with the optional sign, then change the linefeed count by n. The default value for n is the minimum value determined by the -linespace control argument. The linefeed count specifies the number of newline characters (ASCII NL) following each output text line and causes (n-1) blank lines to separate lines of text.

page-define: .pd{m}

Set page definition parameters according the modifier given.

all: .pd; {<d>,<w>}, no break, no substitution

Define the page according to the ordered set of values <d>,<w>. See the individual modifiers with the same letter codes following for additional information. If a value is not given for a parameter, (i.e., its field is blank or null), then its default value is used.

depth: .pdd; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the page depth to n lines. If n is given with the optional sign, then change the current page depth by n. If the resulting page depth is zero or negative, an error diagnostic message is produced. The default value for n is 66.

width: .pwd; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the page width to n columns. If n is given with the optional sign, then change the current page width by n. If the resulting page width is zero or negative, an error diagnostic message is produced. The default value for n is 65.

page-space: .ps; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the formfeed count for the online printer to n. If n is given with the optional sign, then change the formfeed count by n. The default value for n is 1. This control affects only output destined for the online printer. The formfeed count specifies the number of formfeed characters (ASCII FF) sent to the online printer after each page is printed and causes (n-1) blank pages to separate pages of output.

read: .rd; no parameters, no break, no substitution

Read one line from the user_input data stream and process it as an input line. Normal processing resumes with the line following the line containing the read control or as determined by a possible control line read from user_input.

return: .rt; no parameters, no break, no substitution

Terminate processing of the current file and close it. If the current file is an inserted file (see the insert-file control above), resume processing the last suspended file in the insert file stack. If the current file was given in the command line, then begin processing the next file in the input file list. If the input file list is exhausted, then terminate the command normally.

set-reference: .sr{m}

Set variable values and attributes according to the modifer given.

counter: .src; <name> <expr1> {by ±<expr2>}, no break, substitution

<expr1> is evaluated and assigned as the value of <name>. <expr2> is evaluated and assigned as the increment for <name>. If the sign is not given for <expr2>, it is assumed positive. If <expr1> is omitted or cannot be evaluated properly an error diagnostic is produced. If the "by ±<expr2>" clause is omitted, a default value of +1 is assumed. If <name> exists, it must be a user defined counter. If <name> does not exist, it is created as a user defined counter.

mode: .srm; {mode} {<name>,<name>,...}, no break, no substitution

Set the display mode for variables {<name>,<name>,...} to {mode} according to the mode list below. If the variable name list, {<name>,<name>,...}, is not given, then the display mode of the builtin page counter is set. The default value for {mode} is "ar".

mode	Display
ar	arabic numerals (0,1,2,...)
bi	binary numerals (0,1,10,11,100,...)
oc	octal numerals (0,1,2,...,7,10,11,...)
al	lowercase alphabetic (Ø,a,b,...,z,aa,bb,...,zz,...)
au	uppercase alphabetic (Ø,A,B,...,Z,AA,BB,...,ZZ,...)
rl	lowercase roman (Ø,i,ii,iii,iv,v,vi,...)
ru	uppercase roman (Ø,I,II,III,IV,V,VI,...)

variable: .sr, .srv; <name> <expr>, no break, substitution

<expr> is evaluated and assigned as the value of <name>. <name> may be either a user defined variable or a builtin program variable subject to change by the user. If <name> exists and its type does not match the type of <expr>, conversion to the type of

<name> is attempted. If <name> does not exist, it is defined as a user variable with the type of <expr>. If <expr> is omitted or cannot be evaluated properly, or the attempted conversion fails, or <name> is a read-only program variable, an error diagnostic is produced.

space: .sp{m}

Insert blank lines into the output according to the modifer given.

block: .sp, spb: {<#>}, block break, no substitution

Cause a block break processing any pending text as a short line, then, if sufficient space remains on the current page, insert # blank lines. If there is not sufficient space, then begin a new page but do not carry forward any blank lines. A blank or null line in the text has the effect of a ".sp 1" control. The default value for # is 1.

format: .spf; {<#>}, format break, no substitution

Cause a format break processing any pending text as a short line, then, if sufficient space remains on the current page, insert # blank lines. If there is not sufficient space, then begin a new page, print the current text block, and insert # blanks lines. This control form has the effect of a "conditional keep" that assures contiguous white space. The default value for # is 1.

title-block: .tb{m}

Define a heading or caption for the current text block according to the modifier given.

begin: .tb, .tbb {h|c}, no break, no substitution

Suspend text processing, then define a text title block of the type specified according to the input lines following. If the parameter is "h" then define a text heading; if the parameter is "c" then define a text caption. Processing of lines into the title block will continue until the occurrence of a title-block-end control. Input lines for the title block may be text lines, controls, or <title> lines. <title> lines may contain the optional left margin adjustment given before the actual <title> and substitutable variables. Substitution is done when the title is inserted.

end: .tbe; no parameters, no break, no substitution

Stop processing a title block as begun by a preceding title-block-begin control and revert to text processing. If not in title block mode, then ignore the control.

test: .ts; <expr>, no break, substitution

Skip the next input line if the evaluated <expr> is zero, false, or null. If <expr> is omitted, the control is ignored.

text-title: .tt{m}

Define text heading or caption according to the modifier given. The following actions are taken on the parameters for each of the modifiers. If <title> is omitted, then no title line is produced. If <title> consists of one or more occurrences of the same character, then a blank title line is produced. If n is omitted, then <title> is aligned according the value of left indentation at the time of header insertion. If n is given without the optional sign, then <title> is aligned at column n. If n is given with the optional sign, then n is used as a adjustment to the value of left indentation at the time of header insertion. n may not be given unless # is also given. # specifies the number of separating blanks lines for the title line. The blank lines follow a heading line and precede a caption line. The default value for # is 0.

caption: .ttc; {<#>}{<±n>}{<title>}, no break, substitution when inserted

Define a text caption to be appended to the end of the current text block. Multiple occurrences of the text-title-caption control within a text block are accumulated into the caption. Any blank lines specified by # precede the title line. The accumulated number of caption lines is added to the text block size and to the current widow size when the text block is composed into the output page.

heading: .tth; {<#>}{<±n>}{<title>}, no break, substitution when inserted

Define a text heading to be appended to the beginning of the current text block. Multiple occurrences of the text-title-heading control are accumulated into the heading. The heading must be complete before the first text line of the block. Any blank lines specified by # follow the title line. The accumulated number of heading lines is added to the text block size and to the current widow size when the text block is composed into the output page.

translate: .tr; <cd><cd>..., no break, no substitution

The nonblank character "c" is replaced with the character "d" in the composed output. An unpaired "c" at the end of the <variable_field> is treated as though it were paired with a blank. Any number of "cd" pairs may be given (without separating blanks) in the <variable_field> and the "cd" pairs from multiple occurrences of the translate control are accumulated. The translation specified by a "cd" pair may be cancelled only by a translate control giving "cc" in the <variable_field>. Translation of characters to blanks is useful in preserving the contiguous identity of strings during line filling and adjustment. If the <variable_field> is empty, the control is ignored.

type: .ty; {<expr>}, no break, substitution

The string <expr> is evaluated and written to the error_output stream. If <expr> is omitted, a blank line is written.

undent: .un{m}

Adjust the indentation point for the next output line only according to the modifier given. The following actions are taken on the parameter for each of the modifiers. If n is unsigned or has the + sign,

the indentation point is moved toward the associated margin by n columns. If n has the - sign, the indentation point is moved toward the center of the page by n columns. The default value for n is the current associated indentation value.

left: .un, .unl; {<±n>}, format break, no substitution

Adjust the left indentation point.

left-nobreak: .unn; {<±n>}, no break, no substitution

Adjust the left indentation point but do not cause a format break. This control causes the preceding text line (if any) to be padded if the align-both mode is ON.

right: .unr; {<±n>}, no break,

Adjust the right indentation point.

use-reference: .ur; <expr>, no break, substitution

<expr> is subjected to substitution of variables. Substitutable variables delimited with the current special delimiter character are replaced with their current values and the nesting level of special delimiter characters is reduced by one. Variables that are undefined at the time of reference are given the values, zero, null, or false, depending on the required mode.

vertical-margin: .vm{m}

Set vertical margin parameters according the modifier given.

all: .vm; {<t>,<h>,<f>,<b>}, no break, no substitution

Define the vertical margins according to the ordered set of values <t>,<h>,<f>,<b>. See the individual modifiers with the same letter codes following for additional information. If a value is not given for a parameter, (i.e., its field is blank or null), then its default value is used.

top: .vmt; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the top margin to n lines. If n is given with the optional sign, then change the current top margin by n. If the resulting top margin is negative, an error diagnostic message is produced. The default value for n is 4.

header: .vmh; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the header margin to n lines. If n is given with the optional sign, then change the current header margin by n. If the resulting header margin is negative, an error diagnostic message is produced. The default value for n is 2.

footer: .vmf; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the footer margin to n lines. If n is given with the optional sign, then change the current footer margin by n. If the resulting footer

margin is negative, an error diagnostic message is produced. The default value for n is 2.

bottom: .vmb; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the bottom margin to n lines. If n is given with the optional sign, then change the current bottom margin by n. If the resulting bottom margin is negative, an error diagnostic message is produced. The default value for n is 4.

widow: .wi{m}

Change the minimum number of lines to be left or moved when splitting text between pages according to the modifier given.

text: .wi, .wit; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the widow size for text blocks to n. If n is given with the optional sign, then change the text block widow size by n. If the resulting widow size is negative, then an error diagnostic message is produced. The default value for n is 2.

footnotes: .wif; {<±n>}, no break, no substitution

If n is given without the optional sign, then set the widow size for footnotes to n. If n is given with the optional sign, then change the footnote widow size by n. If the resulting widow size is negative, then an error diagnostic message is produced. The default value for n is 1.

wait: .wt; no parameters, no break, no substitution

Read one line from the user_input stream and discard it before proceeding with text processing (also see the read control).

BUILTIN SYMBOLS

This section gives descriptions of the builtin program variables. The format of the title line of each descriptive paragraph is:

Name : mode : default value : controls

where:

Name	is the name of the variable.
mode	is the storage mode; possible value is string, numeric, flag, counter, or function.
default value	is the default value of Name; assigned if no control or control argument specifying a value is given, or if a control with a null variable field is given.
controls	is a blank separated list of controls affecting the value.

Ad, AlignBoth : flag : true : .al
This flag has the value "true" when text lines are being padded to achieve aligned left and right margins.

AlignCenter : flag : false : .al
This flag has the value "true" when text blocks are to have ragged left and right margins.

AlignInside : flag : false : .al
This flag has the value "true" when text blocks are to be aligned on the binding edge of the page.

AlignLeft : flag : false : .al
This flag has the value "true" when the left margin of text blocks is to be aligned.

AlignOutside : flag : false : .al
This flag has the value "true" when text blocks are to be aligned away from the binding edge of the page.

AlignRight : flag : false : .al
This flag has the value "true" when the right margin of text blocks is to be aligned.

Art : flag : false : .bba .bea
This flag has the value "true" when lines are being flagged for artwork.

BlockName : string : "" : .bb .be
The name of the current text block. Inline blocks have a null name.

CallingFileName : string : "" : .if .rt
The entryname of the calling file if Depth is greater than 0.

Ce : numeric : 1 : .bbc #
The number of lines to be composed into an unbreakable, centered text block (if given).

CharsTable : string : collate() : .ch
The translation table for the .chars output file.

Charsw : flag : false : none
This flag has the value "true" if the -chars control argument has been given.

Date : string : date() : none
The current date in the form mm/dd/yy.

Depth : numeric : 0 : .if .rt
The current nesting depth of inserted files.

Device : string : "ascii" : none
The name of the device for which output is being composed. This value is set with the -device or -ball control arguments.

Eq : numeric : 1 : .bbe #
The number of lines to be composed into an equation block.

Eqcnt : counter : 1 : none
The equation reference counter.

ExtraMargin : numeric : 20 if -file/-segment; otherwise 0 : none
The amount of extra left margin to be added to all output lines.

Fi : flag : true : .fi
This flag has the value "true" when text lines are being filled.

FileName : string : member of input file list : none
The name of the input file (from the input file list) currently being processed.

Filesw : flag : false : none
This flag has the value "true" when the -file/-segment control argument is given.

Foot : counter : 1 : .ft .bef
The footnote counter.

FootRef : string : "" : .bbf .bef
The parenthesized string value of the footnote counter as used for footnote numbering.

Fp : numeric: 1 : none
The page number of the first output page to be printed. This value is reset to the value of From at the beginning of each pass.

Fr : flag : true : .ftr
This flag has the value "true" when the footnote counter is being reset to 1 at the beginning of each page and is "false" if the footnote counter is running continuously through the document.

From : numeric : 1 : none
The number of the first output page to print as given by the -from control argument.

Ft : flag : false : .bbf .bef
This flag has the value "true" when text is being composed into a footnote.

Galley : flag : false : none
This flag has the value "true" only when the -galley or -lines options are chosen.

Hyphenating : flag : false : .hy
This flag has the value "true" when hyphenation is being attempted.

In : numeric : 0 : .in .inl
The current value of the left indentation.

IndentRight : numeric : 0 : .inr
The current value of the right indentation.

InputFileName : string : <current input file> : none
The name of the current input file as taken from the input file list given in the command line.

InputLines : numeric : 0 : none
The current line number of the current input file (does not count through inserted files).

Keep : flag : false : .bbk .bek
This flag has the value "true" when the current text block has been designated as a keep block.

LinesLeft : numeric : 54 : .pdd .vmt .vmb .bbf .bef
The number of lines left on the current page available for main body text.

Ll : numeric : 65 : .pdw
The line length, that is, the number of text columns available in output lines.

Lp : numeric : (last page) : none
The number of the last page to be printed.

Ma1 : numeric : 3 : .vmt
The top margin, that is, the number of blank lines between the top of the page and the first header line.

Ma2 : numeric : 3 : .vmh
The header margin, that is, the number of blank lines between the last header line and the first text block.

Ma3 : numeric : 3 : .vmf
The footer margin, that is, the number of blank lines between the last text block (or last footnote) and the last footer line.

Ma4 : numeric : 3 : .vmb
The bottom margin, that is, the number of blank lines between the first footer line and the bottom of the page.

Ms : numeric : 1 : .ls
The line spacing value; 1 = single-space, 2 = double-space, etc.

MultiplePagecount : numeric : 1 : .ps
The number of formfeed characters separating pages in -file mode.

Nl : numeric : 1 : none
The number of the current output line on the page.

NNp : numeric : 1 .br
The page number of the next page to be printed.

NoFtNo : flag : false : .ft .bbf s
This flag has the value "true" when footnote numbering is being suppressed.

Np : numeric : 1 : .br
The number of the current page.

PadLeft : flag : false : none
Obsolete left/right padding switch. It is always "false".

Parameter : string : "" : .if
The value of the string passed via the -parameter control argument or the value passed to an inserted file.

ParamPresent : flag : false : .if <expr>
This flag has the value "true" is Depth is greater than 0 and <expr> was given for the insert-file control.

Passes : numeric : 1 : none
The number of processing passes to perform as given by the -passes control argument.

Pi : numeric : 0 : .bbp
The accumulated number of picture lines reserved.

Pl : numeric : 66 : .pdd
The current page length in lines.

Print : flag : true : none
This flag has the value "true" when the current output line is to be printed.

PrinterSw : flag : true : none
This flag has the value "true" when the -file option has been chosen.

PrintLineNumbers : flag : false : none
This flag has the value "true" when the -number option has been chosen.

Roman : flag : false : .srm
This flag has the value "true" when the page counter is to be displayed in Roman lowercase numerals.

Selsw : numeric : 0 : none
The number of the typeball given by the -ball control argument.

SpecCh : string : "%" : .ch
The current special delimiter character.

Start : numeric : 1 : none
The starting page number as given by the -from and -page control arguments.

Stopsw : flag : false : none
This flag has the value "true" when the -stop option has been chosen.

TextRef : string : "" : .bbf .bef
The superscripted or parenthesized string value of the footnote counter as used for footnote references in text.

Time : numeric : <time> : none
The current time given as the number of seconds since 0000 hours, January 1, 1901.

To : numeric : (last page) : none
The number of the last page to be printed.

TrTable : string : collate() : .tr
The current character translation table.

Un : numeric : 0 : .unl .unn
The value of left undentation.

UndentRight : numeric : 0 : .unr
The value of right undentation.

UserInput : function : internal label : none
The label value of the procedure that substitutes variables with strings read from the user's terminal.

VMargTop : numeric : 3 : .vmt
The top margin, that is, the number of blank lines between the top of the page and the first header line.

VMargHeader : numeric : 3 : .vmh
The header margin, that is, the number of blank lines between the last header line and the first text block.

VMargFooter : numeric : 3 : .vmf
The footer margin, that is, the number of blank lines between the last text block (or last footnote) and the last footer line.

VMargBottom : numeric : 3 : .vmb
The bottom margin, that is, the number of blank lines between the first footer line and the bottom of the page.

Waitsw : flag : false : none
This flag has the value "true" when the -wait option has been chosen.

Widow : numeric : 2 : .wi .wit
The current text widow size.

WidowFoot : numeric : 1 : .wif
The current footnote widow size.

CONSTRUCTING ARTWORK

The artwork feature permits the user to insert certain conventional overstruck character patterns into an input file and to display them as various symbols and lineart features. The feature is invoked by the use of the "artwork" control in a text block. If a text block is designated as an artwork block, any of the conventional overstruck patterns described below are displayed with the closest representation possible for the output device of the intended symbols or lineart feature.

For ASCII devices, the nearest character is chosen from the 95 character graphic set; for devices with plotting capability, a plotted string is generated; for photocomposing devices, a symbol from a special font or an appropriate rule is chosen. The characteristics and capabilities of supported devices are kept in external data segments known as device driver tables. These tables are named <device>.rf_device_table and are discussed in "Device Driver Tables for runoff" below. The runoff program locates the device driver tables by application of search rules.

Artwork blocks are searched for occurrences of the overstruck character patterns that indicate the size, shape, and position of the desired artwork. Any plain text is reproduced at its given location.

Artwork Symbol Conventions

The following characters are syntactically significant if they are overstruck with another character from the set or with a valid size character.

Line art	Math	Meaning
-		element of a horizontal rule
¦	¦	element of a vertical rule or a vertical bar (depending on overstrike pattern)
/	/	element of a +45 degree slant rule or a division sign (depending on overstrike pattern)
\		element of a -45 degree slant rule

(	(	left semi-circle or a left parenthesis (depending on overstrike pattern)
)	)	right semi-circle or a right parenthesis (depending on overstrike pattern)
^		up arrow, diamond top vertex, half-line up, or superscript (depending on overstrike pattern)
v		down arrow, diamond bottom vertex, half-line down, or subscript (depending on overstrike pattern)
<		left arrow or a diamond left vertex (depending on overstrike pattern)
>		right arrow or a diamond right vertex (depending on overstrike pattern)
	[	left bracket
	]	right bracket
	{	left brace
	}	right brace
	X	multiplication sign (one-high math symbol only)
~		vertical or slant rule terminator.
*		horizontal rule terminator.
"		replicator character showing overstrike but having no pictorial meaning.
H	H	half-line control, up or down (depending on overstrike pattern)
S	S	superscript/subscript control (depending on overstrike pattern)
	=	double vertical bar "concatenate" symbol
	o	"bullet" (one-high math symbol only)

If any of the characters of the set in the column labeled Line art is overstruck with another member of the set, it is treated as part of a lineart construction.

If any of the characters of the set in the column labeled Math is overstruck with a numeric or alphabetic character (not part of either set) it is treated as part of a math symbol and the overstrike character is interpreted as the symbol size as follows:

1 - 0	1 through 10 lines
a - z	11 through 36 lines
A - Z	31 through 56 lines

Ambiguous cases are resolved in favor of lineart by the use of the replicator character.

Artwork Source Syntax

The syntax for artwork construction is as follows:

1. "adjust" and "fill" controls should be OFF to preserve element position in an artwork block.
2. Lineart may be contained within math symbols and vice versa.
3. All rules continue through intersections unless they are specifically terminated by an appropriate terminator character.
4. The minimum size of lozenges (flattened diamonds) is four lines. Smaller lozenges will not have their slant sides positioned properly.
5. Unterminated horizontal rules will generate a reported syntax error at the right margin.
6. Unterminated vertical or slant rules will generate a reported syntax error at the end of the artwork block.
7. Vertical positioning of plain text is the responsibility of the source author. Conventional subscript, superscript, and half-line controls are provided for this purpose.
8. The slant line terminators ("7" or "\") must appear to the left (or right) the number of columns one less than the height of the line. For example, the "7" terminating a 5-high right slant line must be 4 lines below and 4 columns to the left of the "7" beginning the slant line.

DEVICE DRIVER TABLES FOR RUNOFF

The runoff program expects to find the artwork characteristics and capabilities for a device in an external static data segment named <device>.rf_device_table. In this context, <device> is the device name, either the default ASCII device or the name given with the -device control argument. The segment is located by application of the users search rules.

The device driver tables are best created by the use of the create_data_segement tool providing data for the following structure:

```
/*        Begin include file rf_device_table.incl.pl1        */

/*        This include file describes the external device driver segment used by
          doc_runoff to drive the target device.  The segment described must be
          named <device>.rf_device_table and may be created with CDS.    */

dcl 1 device_table based (rf_stat$devptr),

    2 devx fixed bin,                    /* device index value */

/* MATH SYMBOL PARTS

    math_symbol_index values
    1 = [, 2 = ], 3 = {, 4 = }, 5 = (, 6 = ), 7 = |, 8 = ||
    9 = vigule/solidus (/), 10 = X, 11 = bullet
   12 = half-line up, 13 = half-line down,
```

```
   14 = superscript, 15 = subscript,
   */

    (2 top (8) char (80),              /* full line top parts */
    2 half_top (8) char (50),          /* half line top parts */
    2 middle (8) char (60),            /* full line middle parts */
    2 bottom (8) char (60),            /* full line bottom parts */
    2 half_bottom (8) char (40),       /* half line bottom parts */
    2 one_high (15) char (80),         /* 1.5 line complete symbols */
    2 other_part (8) char (55),        /* vertical parts for expansion of
                                          multi-line symbols */

/* LINE ART PARTS */

    2 vert_part char (42),             /* vertical line element */
    2 daro char (80),                  /* downward arrowhead */
    2 uparo char (70)) varying,        /* upward arrowhead */
    2 horiz, (                         /* horizontal line */
      3 start char (16),               /* veritcal positioning */
      3 line char (12),                /* one column line element */
      3 term char (16)) varying, (     /* vertical positioning */
    2 laro char (64),                  /* left-pointing arrowhead */
    2 raro char (64)) varying,         /* right-pointing arrowhead */
    2 diamond,                         /* diamond parts */
      3 top char (40) varying,         /* top vertex */
      3 left, (                        /* left vertex */
        4 start char (10),             /* positioning */
        4 body char (45)) varying,     /* actual vertex */
      3 right, (                       /* right vertex */
        4 start char (10),             /* positioning */
        4 body char (45)) varying,     /* actual vertex */
      3 bottom char (50) varying,      /* bottom vertex */
    2 left_slant, (                    /* left slanting line (\) */
      3 start char (20),               /* positioning */
      3 line char (50)) varying,       /* line element */
    2 right_slant, (                   /* right slanting line (/) */
      3 start char (21),               /* positioning */
      3 line char (50)) varying, (     /* line element */
    2 left_circle char (100),          /* left semi-circle */
    2 right_circle char (100)) varying, /* right semi-circle */

/* MISCELLANEOUS STRINGS */

    2 DTAB char (6) varying;           /* direct tab control */

/*        End include file rf_device_table.incl.pl1        */
```

The strings described above are substituted in various combinations and orders for the conventional artwork constructs and the resulting output line is transmitted to the output device in "rawo" mode. The standard system provides five device tables as follows:

Device Name	Device
ascii	The default ASCII terminal device
dtc300s	Data Terminals and Communications 300/S
selecterm	Bedford Computer Systems S75
2741-963	IBM 2741 (EBCDIC)
2741-015	IBM 2741 (Correspondence)

EXAMPLES

This section gives examples of plain text and artwork using runoff.

Plain Text Example

The lines following represent a printed list of a "test.runoff" file.

```
.*  Input file for plain text example
.*
.sp
.tth 1 0 ""TEXT SAMPLE""
.un -5
The runoff command lets the user format text segments through
a variety of controls.  The controls specify such things as:
.sp 2
.in 10
.un 5
1.   Page depth and width (with .pd, .pdd, and .pdw controls).
If not specified by the user, these parameters are given
default values of:
.sp
.in +5
page depth          66 lines
.brf
page width          65 columns
.in 0
.*  End of test.runoff file
```

The same input as formatted:

TEXT SAMPLE

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

1. Page depth and width (with .pd, .pdd, and .pdw controls). If not specified by the user, these parameters are given default values of:

   page depth 66 lines
   page width 65 columns

Artwork Example

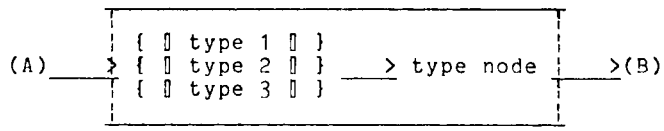
The lines following represent a printed list of an "artwork.runoff" file.

```
.*  Input file for artwork example
.*
.sp
.in 16
.nf
.tth 1 0 ""ARTWORK SAMPLE""
.tth 1 0 ""Section 4 of Flowchart""
.bba
       ¤                                   ¤
       ¶                                   ¶
         ⅜ ⅜ type 1 ⅜ ⅜
ñ¶¤añ¶¤   ¤¤ ⅜ ⅜ type 2 ⅜ ⅜ ñ¤  ¤¤ type node ñ¤    ¤¶¤ñ¶ñ¶¤¤
         ⅜ ⅜ type 3 ⅜ ⅜
       ¤                                   ¤
.bea
```

The same input formatted for an "ascii" terminal.

ARTWORK SAMPLE

Section 4 of Flowchart



The same input formatted for a DTC 300/S terminal.

ARTWORK SAMPLE

Section 4 of Flowchart