To: MTB Distribution

From: E. J. Wallman

Date: 1977 October 11

Subject: New runoff revisited

A design review was held 1977 June 9 on MTB 337, "New runoff implementation". This MTB documents the results of that review and serves as the next step in the formal installation of an advanced text formatting capability on Multics.

The significant items resulting from the review are...

1. The pursuit of the goal of compatibility with BCPL runoff is to be abandoned and an entirely new text formatting program is to be created. This new program is to be named "compose" (with added name and entry "comp") and is to enjoy maximum freedom of design.

2. Concurrent with the release of compose, a source converter (tentatively named "convert_runoff" with "cv_rf" added name and entry) will be released. This tool will convert runoff input files to compose input files.

3. The decision to replace the BCPL implementation of runoff with a truly compatible and more efficient PL/I implementation is tabled for review by Multics Project management. The implementation chosen (if any) should be selected on the basis of the results of compare_ascii tests on the output.

4. Fourteen secondary changes were requested during the review and were accepted for implementation. Some of these are/were already planned.

   - Support graphic set exceptions (like the .chars feature) for all devices including IBM 2741s.
   - Provide an artwork test case file for persons creating device tables for new terminals.

- Accept multi-segment files as input.
- In galley mode, print a footnote header after the footnote to show proper exit from footnote mode.
- Delete the -file and -nocontrol control arguments.
- Change the default extra left margin space from 20 to 0 for output intended for the line printer.
- Change the undocumented debugging control arguments as follows:

  -bugfile, -bf -> -debug_file, -dbf
  -after, -af -> -debug_after, -dba
  -pause, -ps -> -debug_pause, -dbp

- Provide an info file describing the differences between runoff and compose.
- Produce an error diagnostic message for variable references of the form %a b%.
- Show line numbers (for the -number option) in the form "l n" where "l" is the insert file index and "n" is the input line number within the file currently active. Print the list of insert files, showing the index numbers, on user_output after the document output is complete.
- Provide an InputFileDir builtin (similar to &ec_dir) that will contain the pathname of the directory containing the current input file, whose name is in InputFileName.
- Allow dynamic changes of the least word part for hyphenation with an optional parameter on the .hyn control.
- Provide an indent_both (.inb) control for ease in subparagraphing.
- Remove the automatic numbering feature of the -lines option.

The remainder of this MTB is a restatement of MTB 337 reflecting the redirection of the effort. Draft MPM style documentation for compose is attached.

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 for 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.

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 powerful, flexible, and extensible document formatter 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 for installation in the standard system.

---

(1) Registered Trademark, Xerox Corporation.

HIGHLIGHTS OF THE IMPLEMENTATION

- A new control structure and syntax is provided. The new controls have high mnemonic content and generally occur in matched pairs (e.g., "in"/"out", "on"/"off", "begin"/"end", "left"/"right", etc.). As in runoff, a control must be followed by at least one blank (ASCII SP). The following list sumarizes the new controls. (In this list, a slashed-b, "ⱥ", represents a literal blank character.)

.* {<string>}	comment
.~ {<string>}	comment
.al	align
b	both (left/right)
c	center
i	inside
l	left
o	outside
r	right
.bb	block-begin
ⱥ	inline
a {<#>}	artwork
e {<#>}	equations
f {s}	footnote
i	inline
k {<#>}	keep
l {<#>}	literal
n {<name>}	named(1)
p {<#>}	picture(2)
.be	block-end
ⱥ	all
a	art
e	equations
f	footnote
k	keep
l	literal
n	named(1)
p	picture(1)
.br	break
ⱥ	format
b	block
f	format
n {n}	need
p {e|o|i|+n}	page
s {n}{<string>}	skip

---

(1) Not implemented for initial release.

(2) Formatted picture blocks not implemented for initial release.

.cb	change-bars(1)
d	deletion(1)
f	off(1)
n	on(1)
.csd {c}	change-symbol-delimiter
.ctd {c}	change-title-delimiter
.dmp {-all|-user|-mod|<var list>}	dump(1)
.exc <expr>	execute-command
.fb	footer-block(1)
Ø {<+n>} {e|o|a}	begin(1)
b {<+n>} {e|o|a}	begin(1)
e	end(1)
.fl	fill
Ø	default
f	off
n	on
.fl	footer-line
Ø {<#>}{±n}{title>}	all
a {<#>}{±n}{title>}	all
e {<#>}{±n}{title>}	even
o {<#>}{±n}{title>}	odd
.ft	footnote
h	hold(1)
p	paged
r	running(1)
u	unreferenced
.go <name>	go-to
.hb	header-block(1)
Ø {<+n>} {e|o|a}	begin(1)
b {<+n>} {e|o|a}	begin(1)
e	end(1)
.hl	header-line
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
f	off
n {<#>}	on
w <word>	word(1)
.ibl <name>	insert-block(1)
.ifl <name> {<expr>}	insert-file
.ift	insert-footnotes(1)

---

(1) Not implemented for initial release.

.igr <path>	insert-graphic(1)
.in	indent
ø	left
b {±n}	both (left/right)
l {±n}	left
r {±n}	right
.la <name>	label
.ls {±n}	line-space
.pd	page-define
ø {l,w}	all
l {±n}	length
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
.ta	table(1)
b {<name>} {<col def> ...}	define(1)
f {ccc...}	off(1)
n <name> c	on(1)
.tb	title-block(1)
ø {<+n>} {c|h}	begin(1)
b {<+n>} {c|h}	begin(1)
e	end(1)
.tl	title-line
c {<#>}{±n}{title>}	caption
h {<#>}{±n}{title>}	heading
.tre cd..	translate-exceptions
.trf cd..	translate-formatted
.ts {<expr>}	test
.ty {<expr>}	type
.un	undent
ø {±n}	left
l {±n}	left
n {±n}	left-nobreak
r {±n}	right
.ur <expr>	use-reference

---

(1) Not implemented for initial release.

.vm	vertical-margin
Ø {b,f,h,t}	all
b {±n}	bottom
f {±n}	footer
h {±n}	header
t {±n}	top
.wl	widow
Ø {±n}	text
f {±n}	footnote(1)
t {+n}	text
.wrf <path> <ioa_ ctl string>	write-formatted(1)
.wro <path> <iox_ order>	write-order(1)
.wrt <path> {<text>}	write-text
.wt	wait

The reader should note that three controls (fill, hyphenate, and indent) have a "default" form. The formatting parameters associated with these controls have default values that may be changed with command line control arguments, namely, "-nofill", -hyphenate", and "-indent", respectively.

- Input files (single- or multi-segment) are treated as continuous character strings with lengths derived from the bitcounts of the segments. Control arguments (except -arguments) and input file names may be freely intermixed in the command line. All given control arguments apply to all given input file names. Up to 100 input file pathnames may be given for one invocation of the command. 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. Parameters with braces ("{}") are optional and their default value are given in parentheses. The following list summarizes the control arguments.

  -arguments {arg1 ...}, -ag {arg1 ...} ("") (must be last)(1)
  -check, -ck
  -device {name}, -dv {name} ("ascii")
  -exception_graphics, -excep
  -execute "ctl1;...", -ex "ctl1;..." ("")(1)
  -from {n}, -fm {n} (1)
  -galley {n1}{,n2}, -gl {n1}{,n2} (1,end-of-file)
  -hyphenate {n}, -hyph {n}, -hph {n} (3)
  -indent {n}, -in {n} (0)
  -input_file path, -if path (path is required)
  -linespace {n}, -ls {n} (1)
  -noart, -noa
  -nofill, -nof
  -number, -nb
  -number_brief, -nbb

---

(1) Not implemented for initial release.

-output_file {path}, -of {path} ([wd]>input_file.compout)
-page {n1n,n}, -pg {n1n,n} (n is required)
-parameter string, -pm string (string is required)
-pass {n} (1)
-stop, -sp
-to {n} (end-of-file)
-wait, -wt

The following controls will not be in the user documentation:

-debug {n1}{,n2}, -db {n1}{,n2} (1,end-of-file)
causes certain program debugging messages associated with the processing of lines {n1} to {n2} of the input file to be written to user_output. These messages allow the developer to observe the details of the processing of text and controls.

-debug_after {n}, -dba {n} (1)
produce debugging output only after line {n} of the command line input file is processed. This feature permits the skipping of the first N occurences of an insert file in order to debug the (N+1)st occurence.

-debug_file {file}, -dbf {file} (command line file)
produce debugging output only when processing the file {file}. {file} may be either a command line input file or an inserted file.

-debug_pause, -dbp
causes the program to pause after reading each input line in the debug range and to wait for a NL character from user_input. The input characters from user_input are discarded. This feature assures a "clean point" at which the developer can QUIT, invoke probe or debug, and restart the processing.

-meter, -mt
causes the program to record metering data for later display with the "compm" command. The data recorded include elasped time, virtual CPU time, page faults, input lines read, non-blank output lines printed, counts of the number of occurences of each control, and counts of the number of references to each builtin variable.

- 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.

- Page composition is accomplished by means of text blocks. Text is obtained from the input file(s) and is formatted into output lines according to the controls. The formatted output lines are accumulated into a text block until an explicit or implicit text block 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., picture blocks and 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 output 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 attempts to make efficient use of system resources in order to minimize the dollar cost of usage. Continuing development will consider resource efficiency as mandatory.

- The implementation makes extensive use of temporary segments (and the temporary segment manager) and defined areas 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 .ia control encountered. These retained data make it unnecessary to ever rescan an input file when searching for labels. In support of dynamic insert files, 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 symbol 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.

- 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.

- The conventional use of the symbol delimiter character as a reference to the current page number is not supported. The value of the current page number may be obtained only through use of the builtin variable "PageNo". 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:

.fl ||page %PageNo% of %PageNo%||

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

- The implementation provides a galley format option that permits a range of input file line numbers to be specified. 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 provided is defined as unpaged, single column text without running headers and footers and with footnotes immediately following the referencing paragraph.

- The value displayed for the -number control argument may be a compound number of the form "i r". If "i" is present, it is the internal index number of an inserted file whose entryname and pathname are given in a list following the formatted output in the same manner as translator include files. "n" is always the input file line number of the line containing the first printed character of the output line. 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 BCPL runoff and compose.

Missing:

- ^<name> negation of switch control args is not supported.
- The -no_control control argument is not supported.
- The -ball control argument is not supported.
- The -no_pagination control argument is not supported.

Changed:

- New controls are proposed. See mapping below and control descriptions in the attached MPM documentation.
- Line padding is distributed.
- Label names must be unique within an input file.
- 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.
- The -page control argument accepts a list of pages to be printed instead of a new value for the first page number.
- The -segment control argument is changed to -output_file and will accept a pathname for a bulk output file.

- The -chars control argument is changed to -exception_graphics.

Added:

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

-argument(1)	argument array
-change_bars(1)	text change symbols
-check	syntax check mode
-execute(1)	execute given control list
-galley	unpaged output for proofing
-linespace	change default linespace value
-input_file	input file path
-noart	inhibit artwork generation
-output_file	bulk output file path

- 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

---

(1) Not implemented for initial release.

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	.csd	.ce	.bbe n
.ch	.tre	.ds	.ls 2	.ef	.fle
.eh	.hle	.eq	.bbe n	.ex	.exc
.fh	.hlf	.fl	.fln	.fo	.fla
.fr t	.ftp	.fr f	.ftr	.fr u	.ftu
.ft	.bbf/.bef	.gb	.go	.gf	.go
.he	.hla	.lf	.lfl	.ln	.inl
.la	.la	.ll	.bbl n	.ll	.pdw
.m1	.vmt	.m2	.vmh	.m3	.vmf
.m4	.vmb	.ma	.vmt/.vmb	.mp	.ps
.ms	.ls	.na	.all	.ne	.brn
.nf	.flf	.of	.flo	.oh	.hlo
.op	.brp o	.pa	.brp nl±n	.pl	.bbp n
.pl	.pdl	.rd	.rd	.ro	.srm rl
.sk	.brs	.sp	.spb	.sr	.srv
.ss	.ls 1	.tr	.trf	.ts	.ts
.ty	.ty	.un	.unl	.ur	.ur
.wt	.wt	.*	.*	.~	.~

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

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

Name: compose, comp

The compose command is used to prepare formatted documents from raw 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 compose controls are discussed later in this description.

Usage

compose paths {-control_args}

where:

1. paths
   are the pathnames of input files to be processed. The suffix "compin" must be the last component of the input file entrynames; however. the suffix need not be supplied in the command line. If two or more pathnames are specified, they are treated as if compose had been invoked separately for each. Up to one hundred (100) input files may be given with one invocation of the command. Output is produced in the order in which the pathnames are given in the command line. Input files may be either single-segment (SSF) or multi-segment (MSF) files. Output files for very large documents are converted to multi-segment files.

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. Except for "-arguments". control arguments may be freely intermixed with input file pathnames.

   -arguments argi .... -ag argi ...
   When given. this control argument must be the last control argument in the command line. It indicates that all fields following are string values to be placed in the indefinite set of program builtin variables named "CommandArg1" through "CommandArgn" where "n" is the number of such fields. The program builtin variable "CommandArgCount" is set to "n".

   -change_bars {p,l,r,d}. -cb {p,l,r,d}
   enables the generation of text change symbols in the output according to the parameters given. Change symbols are shown in the text margins as determined by controls in the text. The default for change symbol generation is OFF. All the parameters for this control argument are optional but, if any are given, they must appear in the order shown. If any parameter is skipped. its separating comma must still be given. Skipped parameters retain their default values. The parameters are...

   {p} a symbol placement key character and may have the values; "l" for left margin. "r" for right margin. "i" for inside margin. or "o" for outside margin. The default for {p} is "o".

{l} the definition of the text change symbol to be placed to the left of text. It must be of the form "<string>n" where "<string>" is any character string and "n" is the separation from the text. The default value for "<string>" is a vertical bar ("|") and the default value for n" is 2. "n" may not be given unless "<string>" is also given. "<string>" may be given as a conventional artwork symbol (see "Constructing Artwork" later in this description).
{r} the definition of the text change symbol to be placed to the right of text. It must be of the same form as {l} above.
{d} the definition of the text deletion symbol. It must be of the same form as {l} above except that the detault for "<string>" is the asterisk ("*").

-check, -ck
performs syntax checking on the input file(s) by processing all text and controls but does not produce any output. 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 -output_file control argument is given. the default device is taken to be an ASCII line printer. If the -output_file 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}.comp_device_table exists is a supported device (see "Device Driver Tables for compose" later in this description).

-exception_graphics, -excep
flags certain key characters and lines in the output by writing the composed output line to a segment the name of which is formed by replacing the "compin" suffix of the input file entryname with the suffix "compx". In this description, this segment is referred to as the exception-graphics file. Normal output is not affected. Page and line numbers referring to the normal output appear with each flagged line in the exception-graphics file. The default set of key characters is given in the device driver table for the target device and may be modified by the user (see the translate-exceptions control under "Controls" later in this description). The default for this feature is OFF.

-execute "ctl 1;ctl 2;...;ctl n",...;ctl 1;ctl 2;...;ctl n"
The parameter for this control argument is a single quoted, semi-colon separated string of compose controls. The given controls are executed before each given input file and may serve to create an "environment" for composition.

-from {n}, -fm {n}
starts printed output at page {n}. This control argument is mutually exclusive with the -pages control argument. The default value of {n} is 1.

-galley {n1}{,n2}, -gl {n1}{,n2}
produces galley format (unpaged single column text without running headers and footers) output for lines {n1} through {n2} of the input file. 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.

-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 space is in addition to any indentation given with indent-left controls in the text. The default value of {n} is 0.

-input_file <path>, -if <path>
<path> is the name of an input file even though it may have the appearance of a numeric parameter or a control argument.

-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.

-noart, -noa
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.

-nofill, -nof
sets the default fill mode to OFF. If this control argument is not given, the default fill mode is ON.

-number, -nb
prints input line numbers at the left margin of the output. The line numbers have the form "i n" where "i" is the index number of an inserted file. A list of inserted files showing the index numbers is written on user_output after completion of all text processing. The default for this feature is OFF.

-number_brief, -nbb
prints input line numbers at the left margin of the output as for the -number control argument but the list of inserted files is not produced.

-output_file {<path>}, -of {<path>}
directs the composed output to a file instead of to the user's terminal. If {<path>} is not given, then the output for all given input files is written to individual output files the names of which are formed by replacing the "compin" suffix of the input file entrynames with the suffix "compout". If {<path>} is given, then output for all given input files is accumulated in that single bulk output file. The default for this feature is OFF.

-pages n|{n,n} {...}, -pgs n|{n,n} {...}
specifies a blank separated list of selected pages to be printed. Each member of the list must be a single page, {n}, or a range of pages, {n,n}. The page numbers given must steadily increase without duplication. At least one page must be specified. Up to one hundred (100) pages may be specified. This control argument is mutually exclusive with the -from and -to control arguments. The default for this feature is OFF.

-parameter {"string"}, -pm {"string"}
assigns the string value "string" to the builtin variable "Parameter". The default value for "string" is an empty string.

-passes {n}, -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.

-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 {n}. This control argument is mutually exclusive with the -pages control argument. 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 compose input file contains intermixed text and controls. Controls are distinguished from text by their format; ".XXM <variable_field>". See "Preparation of Input Files for compose" later in this description.

2. Summary of text controls See "Preparation of Input Files for compose" later in this description. In this list, ∅ is a literal blank (ASCII SP).

.* {<string>}	comment
.~ {<string>}	comment written to .compx file
.al	align
b	both (left/right)
c	center
i	inside
l	left
o	outside
r	right
.bb	block-begin
∅	inline
a {#}	artwork
e {#}	equations
f {s}	footnote
i	inline
k {#}	keep
l {#}	literal
n {<name>}{,a|r}	named
p {#}	picture
.be	block-end
∅	all
a	artwork
e	equations
f	footnote
k	keep
l	literal

compose

```
   n                                  named
   p                                  picture
.br                                   break
   ø                                  block
   b                                  block
   f                                  format
   n {#}                              need
   p {e|o|#|<+n>}                     page
   s {#} {<string>}                   skip
.cb                                   change-bars
   d                                  deletion
   f                                  off
   n                                  on
.csd {c}                              change-symbol-delimiter
.ctd {c}                              change-title-delimiter
.dmp {-a|-u|-m|<var list>}            dump
.err <string>                         error
.exc <string>                         execute-command
.fb                                   footer-block
   ø {<±n>} {e|o|a}                   begin
   b {<±n>} {e|o|a}                   begin
   e                                  end
.fi                                   fill
   ø                                  default
   f                                  off
   n                                  on
.fl                                   footer-line
   ø {#} {<±n>} {<title>}             all
   a {#} {<±n>} {<title>}             all
   e {#} {<±n>} {<title>}             even
   o {#} {<±n>} {<title>}             odd
.ft                                   footnote
   h                                  hold
   p                                  paged
   r                                  running
   u                                  unreferenced
.go <name>                            go-to
.hb                                   header-block
   ø {<±n>} {e|o|a}                   begin
   b {<±n>} {e|o|a}                   begin
   e                                  end
.hl                                   header-line
   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
   f                                  off
   n {<n>}                            on
   w <word>                           word
.ibl <name>                           insert-block
.ifi <name> {<expr>}                  insert-file
.ift                                  insert-footnotes
.igr <path> {<name>}                  insert-graphic
.in                                   indent
   ø {<±n>}                           left
```

compose

```
   b {<±n>}                                          both
   l {<±n>}                                          left
   r {<±n>}                                          right
.la <name>                                           label
.ls {<±n>}                                           line-space
.pd                                                  page-define
   ø {l.w}                                           all
   l {<±n>}                                          length
   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
.ta                                                  table
   b {<name>} {{f}{a}n{s},{f}{a}n{s},...}            define
   f {ccc...}                                        off
   n <name> c                                        on
.tb                                                  title-block
   ø {<±n>} {c|h}                                    begin
   b {<±n>} {c|h}                                    begin
   e                                                 end
.tl                                                  title-line
   c {#} {<±n>} {<title>}                            caption
   h {#} {<±n>} {<title>}                            heading
.tre {cd..}                                          translate-exceptions
.trf {cd..}                                          translate-formatted
.ts {<expr>}                                         test
.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 {<±n>}                                          text
.wrf <path> <ioa_ arguments>                         write-formatted
.wro <path> <iox_ order>                             write-order
.wrt <path> <text>                                   write-text
.wt                                                  wait
```

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

AlignMode	text alignment mode
ArtMode	artwork block flag

BlockName	the name of the current text block
CallingFileName	name of calling file
CallingLineNo	line number of .ifi in calling file
Date	current date
Device	output device name
Eqcnt	equation reference counter
EqMode	equation mode flag
ExcepOpt	-exception_graphics option flag
ExcepTable	translation table for -exception_graphics
ExtraMargin	extra left margin value
FileName	name of current command line file
FillMode	fill mode flag
Footcnt	footnote counter
FootnoteMode	footnote mode flag
FootReset	footnote reset mode
From	value of -from parameter
Galley	galley mode flag
HeadSpace	blank lines following page or text header
Hyphenating	hyphentation mode flag
Indent	value of left margin indentation
IndentRight	value of right margin indentation
InputDirName	directory containing InputFileName
InputFileName	name of file being processed
InputLineno	line number within InputFileNmae
InsertIndex	current insert file index
KeepMode	keep mode flag
LineNumberOpt	line number option flag
LinesLeft	text lines left on page
LineSpace	line spacing value
NextPageNo	next page number
OutputFileOpt	-output_file option flag
PageLength	current page length
PageLine	line number on current page
PageNo	current page number
PageSpace	count of form feeds between pages
PageWidth	current line length
Parameter	passed parameter value
ParamPresent	passed parameter flag
Pass	-passes value
PictureCount	picture blocks waiting
Print	print flag
StopOpt	-stop option flag
SymbolDelimiter	symbol delimiter character
Time	command invocation time
TitleDelimiter	title part delimiter character
To	-to value
TrTable	text translation table
Undent	undent left margin value
UndentRight	undent right margin value
UserInput	label value for builtin function
VMargBottom	page bottom margin
VMargFooter	footer margin
VMargHeader	header margin
VMargTop	page top margin
WaitOpt	-wait option flag
Widow	current text widow size
WidowFoot	current footnote widow size

Preparation of Input Files for compose

This section discusses the compose control "language" and the preparation of compose 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, line art 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 inserted 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 be formatted text or be a <title> line containing a left margin part, a centered part, and a right margin part (see "Title Lines" later in this description).

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. Footnote reference numbers may be paged (reset to "1" for each page) or running (continuous throughout the document). Footnote reference number insertion may be suppressed either locally for individual footnotes or globally throughout the document. A footnote whose reference number is suppressed does not increment the footnote count. 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 the 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 other controls 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 compose 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 variable fields of other controls by enclosing them between special delimiter characters (see "Symbol Delimiter" and "Embedded Controls" later in this description). Embedded controls have the 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) causes a format break An empty line or a line containing only white space (one or more ASCII SP or HT) causes a block break and generates a blank line in the output. See "Breaks" later in this description.

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 in "Controls" later in this description 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 compose 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-text and widow-footnote controls.

Under certain conditions, the processing of a text block may be "suspended". When a block is suspended, certain parameters and variable values 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.

Two major block types are defined; primary blocks and secondary blocks. Primary blocks are physically separate entities and may not intersect or overlap in any way. Secondary blocks may overlap and may be contained within each other and within primary blocks.

Four primary block types are defined.

inline
: A text block that is composed into the output immediately upon its completion.

named
: A text block that is accumulated for later insertion at a point of the users choice.

page header and page footer
: Text blocks that are inserted at the top and bottom of each output page.

footnote
: A text block containing a conventional text footnote (including a footnote reference number, if any) that is inserted under control of the available footnote processing modes.

Five secondary block types are defined

artwork
: A text block that contains conventional artwork constructs that are converted to line art.

keep
: A text block that is not subject to being split across pages.

picture
: A special form of a keep block that allows following text to be inserted ahead of it.

equation
A text block that may contain <title> lines (see "Title Lines" later in this description).

literal
A text block that may contain text lines that appear to be control lines.

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 end a text block. Examples are indent, undent, page-end-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 "Controls" later in this description).

Fill and Align-both 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 an output 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.

Page Width

The page width is the space available for text in an output line, including all spaces and indentations but not including margins set or implied by the -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 header block	.vmt	4
Between header block and first text line	.vmh	2
Between last text line and footer block	.vmf	2
Between footer block 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 characteristics.

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 the appropriate builtin program variable 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 called 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 text 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.

The text lines in header and footer blocks may be formatted text and/or special lines called <title> lines. The form of the line depends on the choice of control used to define the line. See "Title Lines" below and "Controls" later in this description.

Title Lines

Page headers and footers, text block headings and captions, and equation blocks may contain specially formatted lines known as <title> lines. A <title> contains three distinct parts that are separated with special delimiter characters known as title delimiters. The default title delimiter character is the vertical bar ("|"). The title delimiter character may be changed by the user (see the change-title-delimiter control in "Controls" later in this description).

The 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. Delimiters for parts lying to the right of the last desired part may be omitted. If two successive delimiters are given, the corresponding <title> part is set to an empty (zero length) string. If the <title> consists only of one or more occurrences of the delimiter character only then all parts are empty and the affected line is reset to a blank line.

Text Titles

Text titles are sequences of 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.

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.

Symbol Delimiter

One character of the 95 character ASCII graphic set is designated as a special delimiter for use in expression evaluation. The default character is percent ("%") and may be changed at any time by the user with the change-symbol-delimiter control. Symbol 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.

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 -hyphenate 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). Arithmetic operations yield fixed point decimal results, relational operations yield logical true or false results (with the arithmetic value "-1" representing true and the value "0" representing false), and string operations yield string results.

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)

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:

^* or **	asterisk character
^^ or *^	caret character
^" or *"	double quote character
^b or *b	backspace character (ASCII BS)
^n or *n	newline character (ASCII NL)
^s or *s	blank character (ASCII SP)
^t or *t	horizontal tab character (ASCII HT)
^f or *f	formfeed character (ASCII FF)
^cnnn or *cnnn	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. The "^" character provides a "reconceal" function.

5. Concatenation of strings is indicated by string juxtaposition and is performed from left to right.

6. Substrings are defined as follows; 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 with the -ith character before the end of <string_expression> and the length is set accordingly. If k is negative, the string ends with 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 symbol delimiter characters. (See "Symbol Delimiter" earlier in this description.) If the literal occurrence of the symbol 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 <title>s when they are inserted into the output.

Many of the variables internal to compose 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. Most of the builtin variables may be changed arbitrarily by the user with either an appropriate control or by a reassignment of their values with the set-reference-variable control. However, some builtin variables may not be changed at all or may be changed only in certain controlled ways. These variables are given a "read-only" attribute and are not subject to change with the set-reference-variable control. Those that may change in controlled ways are subject to change only with certain controls.

Two special builtin counters are provided for use in footnote and equation numbering. "Footcnt" 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 "Footcnt" is incremented when the closing block-end-all or block-end-footnote control for a footnote is encountered. Any reference to "Eqcnt" returns the current value and increments the counter. Therefore, its value should be assigned to a user defined variable and that variable 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.

Four special cases of variable reference are defined.

1. If the name of the variable is "UserInput". the value is the string read from the user_input I/O switch. 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 {active_function_args}]", 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 is {<expr>}, then the value is the value of the evaluated <expr>.

4. 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. 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, compose 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 compose. 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 of the form ¦part1¦part1¦part3¦

If any parameter appears without the enclosing less-than, greater-than (<>) characters, it is a literal and must appear as shown. Vertical lines (¦) separate the choices, if any, for literal parameters. Parameters shown within braces ({}) are optional. An elipsis (...) indicates continuation of a parameter string to the extent given in the explanation. If parameters in the template for the <variable_field> of a control are shown separated by a blank column, then they must be separated by white space (ASCII SP or HT) when the control is given. Some controls require commas or other delimiters in their <variable_fields>. If a blank column is not shown, then the white space must not be given.

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 AlignMode builtin variable is set the literal value of the modifier name (e.g., "both" or "center"). Except for align-both, the fill mode may be either ON or OFF. For align-both, the fill mode must be ON. If the fill mode is OFF, overlength lines are not truncated. The default alignment mode is align-both (.alb).

both: .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 control is OFF, this control is mapped into the align-left (.all) control. 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 to 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. Those forms that do not specify a block break define a sub-block within the current block

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 of a block-end-all or block-end-artwork control. This control form is disabled if the -noart control argument is given. If the artwork feature is disabled, no lines are flagged and the block is treated as a normal text block.

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 or be blank. If # is given then suspend the current block mode and accumulate exactly # output equation lines. If # is not given, then change the current block mode to equation mode and continue until the occurrence of a block-end-all 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 default values 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 when leaving footnote mode.

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

If the current block is a named block (see the begin-block-named control below), then suspend processing that block; or, if the current block is a footnote block, then terminate that footnote; and revert to inline block processing. Then (or if the current text block is an inline block), cause a block break, reset any special block modes (art, literal, picture, etc.) in effect and begin a new inline block with the default modes.

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 # output 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-all or block-end-keep control. The break-page and break-need controls are disabled and the break-block control is mapped into break-format 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-all 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 insert-block control as described below. The temporary insert file is created as a uniquely named segment in the users process directory. 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.

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

If # is given, then define an unbreakable picture block of exactly # lines of vertical white space. If # is not given, then accumulate output lines into an unbreakable picture block until the occurrence of a block-end-all 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 vertical white space or a 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. 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, no substitution

If the current block is a named block, then suspend processing that block; or, if the current block is a footnote block, then terminate that footnote; and revert to inline block processing. Then (or if the current text block is an inline block), cause a block break, reset any special block modes (art, literal picture, etc.) in effect and begin a new inline block with the default modes. This form of the block-end control is a "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, 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

Terminate the current footnote block and revert to main body text, queuing the footnote for insertion, saving the footnote text processing parameters, and restoring the main body text processing parameters. Increment the footnote reference counter unless the footnote-unreferenced 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 the footnote mode is not in effect, then ignore the control.

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

Stop accumulating output lines for the unbreakable keep block begun with the block-begin-keep control and reactivate the break-block, 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 control and

the block-end-all control are the only controls 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 processing 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 the picture mode is not in effect. then ignore the control.

break: .br{m}

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

block: .br. .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. In some processing modes (e.g., keep mode and picture mode) that do not allow termination of a text block, this control is mapped into a break-format control.

format: .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 page break, no substitution

If the number of available text lines left on the page is less than # then terminate the current page and begin the next sequential page Any pending text is deferred to the new page. If the number of available text lines on the page is grater than or equal to #, then continue composing output into the current page. The default value for # is 1.

page: .brp; {e|o|#|<±n>}, block and page breaks. 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. No separating blank pages are produced. If no parameter is given, then the page number for the new page is the next sequential page number.

skip: .brs; {#} {<string>}, block and page breaks. 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-bars: .cb{m}

If the -change_bars control argument has been given, then place text change and deletion symbols as determined by the modifer given and as specified in the control argument (see -change_bars control argument under "Usage" earlier in this description). If the -change_bars control argument has not been given, then ignore the control.

deletion: .cbd; no parameters, no break, no substitution

Place a text deletion symbol in the appropriate margin.

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

If change bar generation is ON, then stop generating change bars; otherwise, ignore the control.

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

Begin placing text change symbols in the appropriate margins.

change-symbol-delimiter: .csd; {c}, no break, no substitution

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

change-title-delimiter: ctd; {c}, no break, no substitution

Change the special delimiter character used to delimit <title> parts to "c". The title delimiter character previously defined (including the default title delimiter character) is treated as a normal character. If "c" is omitted. the title delimiter character reverts back to the default title delimiter character. The default title delimiter character is "|".

comment: .*; {<rest of line>}, no break, no substitution

A comment line having no effect on any output.

comment: .~; {<rest of line>}, no break, no substitution

A comment with no effect on normal output. The entire control line is written to the exception-graphics file if the -exception_graphics control argument is given.

dump: .dmp; {-a|-u|-m|<variable list>}, no break, no substitution

Display the names, attributes, and values of program variables according to the parameter given. If no parameter is given or if the parameter is "-a", then display all program variables. If the parameter is "-u", then display only user defined variables. If the parameter is "-m", then display only those variables whose values have changed since the last ocurrence of a dump control. If the parameter a <variable list> containing a blank separated list of variable names, then display only the named variables. The display shows the variable name, the storage type, and the value. Builtin and user defined variables are displayed in two separate sorted blocks.

error: .err; <string>, no break, no substitution

"<string>" is prefixed with the name of the current input file and the current line number within that file and is entered into the error diagnostic message table

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" earlier in this description), does not extend past the right margin. The default for this mode is normally ON but may be changed to OFF with the -nofill control argument.

default: .fi; no parameters, format break, no substitution

Set the fill mode to the default value established by the command line (see the -nofill control argument under "Usage" earlier in this description).

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

Set the fill mode OFF.

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

Set the fill mode ON.

footer-block: .fb{m}

These controls permit the definition of an arbitrarily formatted page footer block. A formatted footer block may contain <title> lines, formatted text, artwork, and many other features available in normal text blocks.

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

Cancel the footer block of the type specified by the second parameter and begin a new formatted footer block of the same type. All text lines, <title> lines, and controls encountered until the ocurrence of a footer-block-end or block-end-all control are considered part of the footer block. If the optional parameter <±n> is given without the optional sign, it is the column number for the left alignment of the block. If it is given with the optional sign, then it is an adjustment to the current left indentation point. The default value for <±n> is 0. If the second parameter is "e", then only the footer block for even pages is defined; if it is "o", then only the footer block for odd pages is defined; if it is "a", then the footer blocks for both even and odd pages are defined. The default value for the second parameter is "a". Any <title> lines given are subject to substitution of variables when the block is inserted into the output.

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

Stop processing input lines into the footer block specified by a previously given footer-block-begin control and return to normal text processing. If not processing footer block lines, then ignore the control.

footer-line: .fl{m}

Define page footer lines according to the modifier given. The following actions are taken on the parameters for each of the modifiers. If no parameters are given, then the current footer block is cancelled. 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 available footer line, then intervening null lines are inserted. If # is less than or equal to the current size of the footer block, then <title> redefines line # of the current footer block. If <±n> is given without the optional sign, it is the column number for left alignment of the footer line. If <±n> is given with the optional sign, it is an adjustment to the current left indentation value. The default value for <±n> is 0. <±n> may not be given unless # is also given. 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. Null lines are not printed. If <title> consists only of one or more occurrences of the title delimiter character, then line # of the footer block is replaced with a blank line. Footer block lines are numbered from the bottom up. Default footer blocks are empty.

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

Define footer lines as discussed above for all pages.

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

Define footer lines as discussed above for even pages only

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

Define footer lines as discussed above for odd pages only.

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 insert-footnote control or, by default, at the end of the document. The footnote counter runs continuously until reset by another footnote control.

paged: .ftp; {#}, 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.

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

Insert footnotes as they appear and on the page of their reference. The footnote counter is reset to # and runs continuously until reset. 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, footnote-paged, or footnote-running controls.

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 line used is the first ".la <name>" line in the file. If <name> is not a label defined in the file, an error diagnostic message is produced, and processing proceeds starting with the line following the ".go <name>" line. If <name> is defined, text processing resumes with the ".la <name>" line selected.

header-block: .hb{m}

These controls permit the definition of an arbitrarily formatted page header block. A formatted header block may contain <title> lines, formatted text, artwork, and many other features available in normal text blocks.

begin: .hb, .hbb; {<±n>} {e|o|a}, no break, no substitution

Cancel the header block of the type specified by the second parameter and begin a new formatted header block of the same type. All text lines, <title> lines, and controls encountered until the ocurrence of a header-block-end or block-end-all control are considered part of the header block. If the optional parameter <±n> is given without the optional sign, it is the column number for the left alignment of the block If it is

given with the optional sign, then it is an adjustment to the current left indentation point. The default value for <±n> is 0. If the second parameter is "e", then only the header block for even pages is defined; if it is "o". then only the header block for odd pages is defined; if it is "a". then the header blocks for both even and odd pages are defined. The default value for the second parameter is "a". Any <title> lines given are subject to substitution of variables when the block is inserted into the output.

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

Stop processing input lines into the header block specified by a previously given header-block-begin control and return to normal text processing. If not processing header block lines, then ignore the control.

header-line: .hl{m}

Define page header lines according to the modifier given. The following actions are taken on the parameters for each of the modifiers. If no parameters are given, then the current header block is cancelled. 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 available header line, then intervening null lines are inserted. If # is less than or equal to the current size of the header block, then <title> redefines line # of the current header block. If <±n> is given without the optional sign, it is the column number for left alignment of the header line. If <±n> is given with the optional sign, it is an adjustment to the current left indentation value. The default value for <±n> is 0. <±n> may not be given unless # is also given. 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. Null lines are not printed. If <title> consists only of one or more occurrences of the title delimiter character, then line # of the header block is replaced with a blank line. Header block lines are numbered from the top down. Default header blocks are empty.

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

Define header lines as discussed above for all pages.

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

Define header lines as discussed above for even pages only.

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

Define the header for footnotes. For this modifier only, any given value of # is forced to 1. 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. The footnote header line is preceded and followed by a single blank line.

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

Define header lines as discussed above for odd pages only.

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. Horizontal tabulation in the output is enabled with the horizontal-tab-on control that associates a named tab stop pattern with a specific character. During composition, each text character is checked against the set of active horizontal tab characters. If a match is found, the fill string for the next tab stop column of the associated pattern is inserted to fill the space up to that tab stop column.

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

Define pattern <name>, setting tab stops at columns (#,#,#,..). 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 (ASCII SP) characters. If no tab stop columns are 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. If fill mode is ON, it is suspended until horizontal tabulation is turned off with a horizontal-tab-off control. When horizontal tabulation is enabled for character <c> and pattern <name>, then every occurrence of <c> in an output line is replaced with white space up to the column given by the first tab stop value in <name> that is greater than the column in which <c> is found.

hyphenate: .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) and set the least word part size to the default value as established by the -hyphenation control argument (see the -hyphenate control argument under "Usage" earlier in this description).

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

Set the hyphenation mode OFF

on: .hyn; {n}, no break, no substitution

Set the hyphenation mode ON. The optional parameter, {n}, is the size of the least word part to be acceptable for hyphenation.

word: .hyw; {-}<hy-phen-ated>, no break, no substitution

Add the word <hyphenated> to the internal special hyphenation table with the hyphenation points indicated by the hyphens. If no hyphens are given, then the word is not subject to hyphenation. If the optional leading hyphen is given (with or without embedded hyphens), then the word is removed from the internal special hyphenation table and hyphenation of that word reverts to the hyphenation dictionary. The internal special hyphenation table may contain up to 100 entries and is searched before any reference to the hyphenation dictionary.

indent: .in{m}

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

both: .inb; {<±n>}, format break, no substitution

If <±n> is given without the optional sign, then set both the left and right indentation points to <n> columns inward from their respective margins. If <±n> is given with the optional sign, then change the current left and right indentation points by <n> columns. Positive values for <±n> cause inward movement. The default value for <±n> is 0. Any value that results in a zero or negative effective line length will produce an error diagnostic message The indentation points will never be set beyond their respective margins. The left margin is determined by the physical characteristics of the output device and the -indent control argument (see "Usage" earlier in this description) and the right margin is determined by the page-define-width control (discussed later in this section).

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. 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. The default value for <±n> is 0. Any value 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 left margin is determined by the physical characteristics of the output device and the -indent control argument.

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. 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. The default value for <±n> is 0. Any value 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 right margin is determined by the page-define-width control (discussed later in this section).

insert-block: .ibl; <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).

insert-file: .ifi; <pathname> {<expr>}, no break, no substitution

Suspend processing of the current input file, open the file <pathname>.compin and begin processing with line 1 of that file. <pathname> may be an absolute or relative pathname and the file is located by application of search rules. <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). Processing of the inserted file continues until the occurrence of a return control or an end-of-file condition in detected When the processing of the inserted file is complete, processing of the suspended input file is resumed with the line following the line containing the insert-file control.

insert-footnotes: .ift; no parameters, no break, no substitution

Insert all pending footnotes and reset the footnote counter to 1.

insert-graphic: .igr; <path> {<name>}, no break, no substitution

Suspend processing the current input file and insert the Permanent Graphic Structure named "<name>" contained in "<path>". The default for "<name>" is <path> with the "pgs" suffix removed. (See "Multics Graphics System", Order No. AS40, for a discussion of Permanent Graphic Structures.)

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>. If the -linespace control argument has been given, then the default value for <±n> is the value given with the control argument. If the -linespace control argument has not been given, then the default value for <±n> is 1. 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 to the modifier given.

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

Define the page according to the ordered set of values <l>,<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.

length: .pdl; {<±n>}, no break, no substitution

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

width: pdw; {<±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 I/O switch 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,...)
hx	hexadecimal numerals (0,1,2, ..,D.E,F,10,11..)
oc	octal numerals (0,1,2,....7,10,11,...)
al	lowercase alphabetic (ϐ.a,b,....z,aa.ab, ,zz,...)
au	uppercase alphabetic (ϐ.A.B,. .,Z,AA AB,...,ZZ,...)
rl	lowercase roman (ϐ.i,ii,iii,iv,v.vi,...)
ru	uppercase roman (ϐ 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 (if not in artwork mode) has the effect of a .spb 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 insert # blank lines into the text block A blank or null line within an artwork block has the effect of a ".spf 1" control. The default value for # is 1.

table: .ta{m}

Define and control formatted table mode according to the modifier given. Up to 10 table formats may be defined but only one may be in effect at any one time. Formatted table mode is enabled with the table-on control that specifies a named table format to be used. A table format may contain up to 10 columns (see table-define control following). When in formatted table mode, each input text line must begin with a single decimal digit (e.g. "1Test line."). The digit indicates the column in which the text is to be composed. By convention, the digit "0" indicates the tenth column of the pattern. In formatted table mode, an input text line given without the digit will cause an error diagnostic message.

define: .tab; <name> {<l>,<w>{<f>}{<a>}:...}: no break, no substitution

Define table format <name>, specifying table columns with the column definition fields "l,w{<f>}{<a>}". Up to 10 columns may be defined for a table format. Each column definition field may contain four parameters; the column left margin, <l>; the column width, <w>; an optional column fill mode, <f>; and an optional column alignment mode, <a>. The column left margin values, <l>. must be given in steadily increasing order. The column width value must be given and must be separated from the column left margin value by a comma. If the character immediately following a column width value is not a colon (":") or newline character, then that character is the column fill mode, <f>, and may have values "n" for unfilled or "f" for filled. The default value for <f> is "f". If the character immediately following a column fill mode is not a colon or newline character, then that character is the column alignment mode. <a>. and may have the values "b" for both, "c" for centered, "l" for left, "r" for right. "i" for inside, or "o" for outside. The column alignment mode may not be given unless the column fill mode is given. The default value for <a> is "b". If no column definitions are given, the table format entry for <name> is cancelled. If no <variable_field> is given, all table formats are cancelled

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

Disable formatted table mode and return to normal text processing. If not in formatted table mode, then ignore the control.

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

Suspend normal text processing and enable formatted table mode with format <name>. If <name> is not given or is not a defined table format, then an error diagnostic message is produced.

compose

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.

title-block: .tb{m}

These controls permit the definition of an arbitrarily formatted text block header or caption block. A formatted title block may contain <title> lines, formatted text, artwork, and many other features available in normal text blocks.

begin: .tb. .tbb; {<±n>} {c|h}, no break, no substitution

Cancel the title block of the type specified by the second parameter and begin a new formatted title block of the same type. All text lines <title> lines, and controls encountered until the ocurrence of a title-block-end or block-end-all control are considered part of the title block. If the optional parameter <±n> is given without the optional sign, it is the column number for the left alignment of the block If it is given with the optional sign, then it is an adjustment to the current left indentation point The default value for <±n> is 0. If the second parameter is "h", then a text header block (ahead of the text) is defined; if it is "c", then a text caption block (after the text) is defined. The default value for the second parameter is h". Any <title> lines given are subject to substitution of variables when the block is inserted into the output.

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

Stop processing input lines into the title block specified by a previously given title-block-begin control and return to normal text processing. If not processing title block lines, then ignore the control.

title-line: .tl{m}

Define text title lines according to the modifier given. The following actions are taken on the parameters for each of the modifiers. If # is given, it specifies the number of blank lines to follow (for headers) or precede (for captions) the title line. The default value for # is 0. If <±n> is given without the optional sign, it is the column number for left alignment of <title>. If <±n> is given with the optional sign, it is an adjustment to the current left indentation value. The default value for <±n> is 0. <±n> may not be given unless # is also given. If <title> is omitted, then line # is replaced with a null line and the original numbering of lines in the title block is not changed. Null lines are not printed. If <title> consists of one or more occurrences of the title delimiter character, then line # of the title block is replaced with a blank line. Text header block lines are numbered from the top down; text caption block lines are numbered from the bottom up. Text header blocks are automatically inserted into the output ahead of the text block in which they are defined. Text caption blocks are automatically inserted into the output after the text block in which they are defined. Text title blocks are cancelled upon insertion. Default text title blocks are empty.

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

Define text caption lines as discussed above.

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

Define text header lines as discussed above.

translate-exceptions: .tre; <cd><cd>.... no break, no substitution

Modify the translation table used in the -exception_graphics feature by adding the <cd> pairs to the default table for the target device. See the -exception_graphics control argument under "Usage" earlier in this description. If the -exception_graphics option is given, then every occurrence of any "c" in the output is flagged by replacing it with the corresponding "d" and writing the output line to the exception-graphics file 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 exception-graphics file. If all "c"s in an output line are paired with blanks the line is not written to the exception-graphics file. If the pair <cc> is given, then "c" is effectively removed from the translation table. The default exception graphics translation table is given in the device table for the target device. Any number of "cd" pairs may be given (without separating blanks) in the <variable_field> and the "cd" pairs from multiple occurrences of the control are accumulated.

translate-formatted: trf; <cd><cd>..., no break, no substitution

The nonblank character "c" in the input 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 control are accumulated. The translation specified by a "cd" pair may be cancelled only by a translate-formatted 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 I/O switch. 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 value of the current associated indentation value.

compose

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, no substitution

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. The evaluated <expr> is then treated as an input line.

vertical-margin: vm{m}

Set vertical page margins 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.

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

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

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 or greater than the current page length, 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 or greater than the current page length, then an error diagnostic message is produced. The default value for <±n> is 1.

write: .wr{m}

These controls permit writing data to arbitrary files for later use either by compose or by other programs. They will find use in documentation systems involving compose where supplementary, text dependent information such as indices and glossaries are required. For all the forms of the control, <path> is required and is the pathname of the file into which data is to be written. If <path> does not exist, it is created. If <path> does exist, it is truncated when it is first opened during an invocation of compose. All attachments of <path> are made through the vfile_ I/O module.

formatted: .wrf; <path> <ctl_string>{,argi,...}, no break, no substitution

<ctl_string> is required and is a string of formatting controls acceptable to the ioa_ subroutine (see MPM Subroutines, Order No AG93, for a discussion of ioa_). No check is made for the validity of a control string. The optional arguments, argi (possibly required by ctl_string) are all string values, either literals or substituted values of variables.

order: .wro; <path> <io order>, no break, no substitution

<io order> is required and is an iox_ order acceptable to the io_call command (see MPM Commands and Active Functions, Order No AG92. for a discussion of the io_call command). Since all attachments are made through the vfile_ I/O module, only those orders supported by vfile_ may be given. No check is made for the validity of an order.

text: .wrt; <path> {<text>}, no break, no substitution

<text> (with an added newline character) is written to <path>. If <text> is omitted only the newline character is written and the result is a null line.

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 assigned to Name 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 and control arguments affecting the value.

AlignMode : string : "both" : .alb .alc .ali .all alo .alr
The current text alignment mode. The possible values are "both", "left" "right", "inside", "outside", and "center".

ArtMode : flag : false : .bba .bea -noart
True when lines are being flagged for artwork; otherwise, false.

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

CallingFileName : string : "" : ifi .rt
The name (no suffix) of the previous file in the insert file stack. If the current file is from the command line input file list, this variable has a null value.

CallingLineNo : numeric : 0 ; .ifi .rt
The line number of the insert-file control in the previous file in the insert file stack If the current file is from the command line input file list, this variable has a zero value.

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

Device : string : "ascii" : -device
The name of the device for which output is being composed This value may be set with the -device control argument.

Eqcnt : counter : 1 : .bbe .bee
The equation reference counter.

EqMode : flag : false : .bbe .bee
True if processing text lines in equation mode; otherwise, false.

ExcepOpt : flag : false : -exception_graphics
True if the -exception_graphics control argument has been given; otherwise false.

ExcepTable : string : depends on device : tre -device
The translation table for the -exception_graphics option.

ExtraMargin : numeric : 0 : -indent
The amount of extra left margin to be added to all output lines. This value may be set with the -indent control argument

FileName : string : entry in input file list : none
The name of the command line input file (from the command line input file list) currently being processed.

FillMode : flag : true : fi .fif .fin -nofill
True when in fill mode; otherwise false. The default value for this flag may be changed with the -nofill control argument.

Footcnt : counter : 1 : .bbf .bef .fth .ftp ftr .ftu
The footnote counter.

FootnoteMode : flag : false : .bbf .bef
True when processing a footnote; otherwise, false.

FootReset : string : "paged" : .fth .ftp .ftr .ftu
This variable indicates the mode for footnote numbering. It may have the values "paged" if footnote numbers are being reset to 1 at the top of each page, "running" if footnote numbers are running continuously. "unref" if footnote numbers are not being used, or "hold" if footnotes are being held for later insertion and their numbers are running continuously.

From : numeric : 1 : -from
The number of the first output page to print as given by the -from control argument.

Galley : flag : false : -galley
True when the -galley control argument is given; otherwise. false.

HeadSpace : numeric : 0 : .vmt .vmh .tlh
The number of blank lines inserted in the immediately preceding page header margin or text header margin. The separating line count, "#" given with the title-block-begin or title-line-header controls is considered a margin value.

Hyphenating : flag : false : hy .hyn .hyf -hyphenate
True when in hyphenation mode; otherwise, false.

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

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

InputFileDir : string : none : .ifi .rt
The pathname of the directory containing InputFileName.

InputFileName : string : none : .ifi .rt
The name of the file currently being processed.

InputLineno : numeric : 0 : none
The current line number in InputFileName

InsertIndex : numeric : 0 : .ifi .rt
The index number of the current file in the insert file stack. If the current file is from the command line input file list, this variable has a zero value. The value of this variable is used for "i" for the -number or -number_brief features.

KeepMode : flag : false : .be .bbk .bek
True when processing a keep block; otherwise, false.

LineNumberOpt : flag : false : -number -number_brief
True when either the -number or -number_brief control arguments are given; otherwise, false.

LinesLeft : numeric : 54 : pd pdl .vm .vmf .vmb .bbf .bef
The number of lines left on the current page available for text.

LineSpace : numeric : 1 : .ls -linespace
The line spacing value; 1 = single-space. 2 = double-space. etc. The default value may be changed with the -linespace control argument.

NextPageNo : numeric : 1 : .brn .brp .brs
The page number of the next page to be printed.

OutputFileOpt : flag : false : -output_file
True when the -output_file control argument is given; otherwise, false.

PageLength : numeric : 66 : .pd .pdl
The current page length in lines.

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

PageNo : numeric : 1 : .brn .brp .brs
The number of the current page.

PageSpace : numeric : 1 : ps
The number of formfeed characters separating pages in output for the online printer.

PageWidth : numeric : 65 : .pd .pdw
The page width, that is. the number of text columns available in output lines.

Parameter : string : "" : .ifi -parameter
The value of the string passed to an inserted file. The initial value may be changed with the -parameter control argument.

ParamPresent : flag : false : .ifi <expr>
True if <expr> is given in an insert-file control; otherwise, false.

Passes : numeric : 1 : none
The number of processing passes remaining to be performed (including the current pass). The initial value may be changed with the -passes control argument. Output is produced only when the value is 1.

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

Print : flag : true : -galley -from -to -pages
True when the current line is to be printed; otherwise, false. This variable is controlled by the selection of lines or pages to be printed with the -galley, -from, -to, or -pages control arguments.

StopOpt : flag : false : -stop
True when the -stop control argument is given; otherwise, false.

SymbolDelimiter : string : "%" : .csd
The current symbol delimiter character.

Time : numeric : <time of day> : none
The time of day at command invocation in the form "hh:mm:ss".

TitleDelimiter : string : "|" : .ctd
The current title part delimiter character.

To : numeric : (last page of input file) : -to
The number of the last page to be printed as given with the -to control argument. If the -to control argument is not given, the value is -1.

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

Undent : 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 internal procedure that is used during substitution of variables to obtain a character string from the user_input I/O switch.

VMargBottom : numeric : 4 : .vm .vmb
The bottom margin, that is, the number of blank lines between the footer block and the bottom of the page.

VMargFooter : numeric : 2 : .vm .vmf
The footer margin, that is, the number of blank lines between the last text block and the footer block.

VMargHeader : numeric : 2 : vm .vmh
The header margin, that is, the number of blank lines between the header block and the first text block

VMargTop : numeric : 4 : .vm .vmt
The top margin, that is, the number of blank lines between the top of the page and the header block.

WaitOpt : flag : false : -wait
True when the -wait control argument is given; otherwise, false.

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 line art features. The feature is invoked by the use of the block-begin-artwork and block-end-artwork controls in a text block. Text lines that fall within the scope of these controls are flagged as artwork lines and any of the conventional overstruck patterns described below are displayed with the closest representation possible for the output device of the intended symbols or line art 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>.comp_device_table and are discussed in "Device Driver Tables for compose" below. The compose program locates the device driver tables by application of search rules.

Artwork lines 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.

In this section, the word "rule" refers to a typographic rule. that is, a line of given length, thickness, and orientation.

Artwork Symbol Conventions

Two sub sets of the 95 character ASCII graphic set are defined; the "line art" set and the "math symbol" set. Members of the line art set are syntactically significant if they are overstruck with another character from the set and members of the math symbol set are syntactically significant is they are overstruck with a valid size character (see the discussion of size characters following).

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, or upward movement (depending on overstrike pattern)
v	v	down arrow, diamond bottom vertex, or downward movement (depending on overstrike pattern)
<	<	left arrow, a diamond left vertex, or leftward movement (depending on overstrike pattern)
>	>	right arrow, a diamond right vertex. or rightward movement (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		half-line control. up or down (depending on overstrike pattern)
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 in the line art set is overstruck with another member of the set. it is treated as part of a line art construction.

If any of the characters in the math symbol set 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
a - z	11 through 36
A - Z	31 through 56

The four movement "symbols" perform "micropositioning" and the size character represents the count of increments to be moved.

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

Artwork Source Syntax

The syntax for artwork construction is as follows:

1. "align-both" and "fill" modes should be OFF to preserve element position in an artwork block.
2. Line art 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. The movement "symbols" 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 terminator for a 5-high right slant line must be 4 lines below and 4 columns to the left of the beginning of the slant line.

DEVICE DRIVER TABLES FOR COMPOSE

The compose program expects to find the artwork characteristics and capabilities for a device in an external static data segment named <device>.comp_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 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 comp_device_table.incl.pl1     */

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

dcl 1 device_table.

    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).                 /* vertical 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 comp_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
2741e	IBM 2741 (EBCDIC)
2741c	IBM 2741 (Correspondence)

EXAMPLES

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

Plain Text Example

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

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

The same input as formatted:

TEXT SAMPLE

The compose 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

compose

Artwork Example

The lines following represent a printed list of an "artwork.compose" file. In this example, the symbol "<BS>" in the comment lines stands for the ASCII backspace character.

```
.*  Input file for artwork example
.*
.spb
.inl 13
.fif
.tlh 1 0 ||ARTWORK SAMPLE||
.tth 1 0 ||Section 4 of Flowchart||
.bba
.*          -<BS>"                                  -<BS>*
.*          ¥                                       ¥
.*          "                                       *
.*          "<BS>|                                  "<BS>|
.*          ¥                                       ¥
.*          "                                       *
.*
.*            3<BS>{      3<BS>=
.*              | 3<BS>=    | 3<BS>}
.*              ¥ ¥         ¥ ¥
.*              3 3 type 1 3 3
.*  ^<BS>H
.*  |(<BS>"
.*  |v<BS>H                                                  ^<BS>H
.*  |  ^<BS>H                                               |-<BS>"
.*  |  |)<BS>"                  3<BS>=                      |    -<BS>><BS>*
.*  |  |-<BS>"                  | 3<BS>}                    |    |(<BS>"
.*  |  |    -<BS>><BS>*        |  ^<BS>H                   |    |v<BS>H
.*  |  |    |v<BS>H            |  |-<BS>"                  |    |  ^<BS>H
.*  |  |    |  3<BS>{          |  |    -<BS>><BS>*         |    |  |)<BS>"
.*  |  |    |  | 3<BS>=        |  |    |v<BS>H             |    |  |v<BS>H
.*  ¥  ¥    ¥  ¥ ¥             ¥  ¥    ¥                    ¥    ¥  ¥  ¥
.*  H(MAH)"   *M 3 3 type 2 3 3 H"   *M type node H"    *(MBH)M
.*          3<BS>{      3<BS>=
.*            | 3<BS>=    | 3<BS>}
.*            ¥ ¥         ¥ ¥
.*            3 3 type 3 3 3
.*       -<BS>"<BS>|<BS>^                    -<BS>*<BS>|<BS>^
.*        ¥                                   ¥
.*        *                                   *
```

The same input formatted for an "ascii" terminal.

ARTWORK SAMPLE

Section 4 of Flowchart