1. Bitmap representation(*width,height,side,top*)
raster-code-sequences

The data defining a bitmap in basic raster form is introduced by a group of four arguments enclosed in parentheses. These values are in device pixels and the raster information must be strictly in accordance with them. The first two arguments are the *width* and *height* of the bitmap. There are no default values for these. The second two arguments specify a positional displacement for placing the image, to the right and down (respectively) from the reference position. The default values for these are zero.

The raster information, which starts on the following line, consists of *sequences*, of which there must be precisely *height* in number. Each sequence accounts for one row of dots, *width* in number, starting with the top row of the bitmap and continuing downwards.

In general, each character in a sequence accounts for six dots in the representation, according to the position of the character code in the ASCII collating sequence based at 'o' (character zero, code 48). So 'o' itself represents six consecutive white points and 'o' (lower-case o, code 111) - which is 63 on from 'o' - represent six consecutive black points.

The most significant bit in the binary representation of the 6-bit value corresponds to the leftmost dot and bit set denotes a black point. The characters in a sequence correspond to successive groups of six dots starting at the left of the row.

To economise on the coding of long runs of similar values, the symbols '#' (code 35) to ',' (code 44) may be used to represent runs of 12,18,...,66 white dots; the symbols 'p' (code 112) to 'v' (code 118) may be used to represent runs of 12,18,...,48 black dots; and symbols 'w' (code 119) upwards may be used to indicate that the basic 6-bit code defined by the immediately following symbol is to be repeated 3,4,.... times.

The number of characters in each sequence is, at a maximum, the smallest number which accounts for a multiple of 6 which is greater than or equal to the bitmap width. The binary values for any excess dots accounted for by the last character should be zero. However, trailing zeros in the binary representation of any row need not be accounted for, provided that the sequence is terminated by a space or format effector. If sufficient characters are present to account for the complete width of the bitmap, there may be such a terminator but need not be; that is, the sequences can simply be run together.

Any row which is identical to the preceding row for this bitmap may be represented by a '"' (for ditto). No terminator is required before or after this symbol.

A sequence may be continued over more than one source line by preceding the line-end by a hyphen (minus-sign).

There must be at least one printing character in each sequence.

2. Graphics character definition(*width,height,side,top*) G(.....)
[*graphics commands*]

This form is introduced by a group of four arguments in parentheses with the same significance as for the raster form, followed on the same line by the letter G (for Graphics). The G may be accompanied by the same arguments used for the G directive (*scale,Xzero,Yzero*). This header is followed by a sequence of graphics commands enclosed in scope brackets ([] or {}). The interpretation of the commands is the same as the interpretation of commands in graphics mode, within a virtual page of the dimensions given by *width* and *height*.

All of the main graphics commands can be used in this form of character definition, but not the text placement or Bitmap commands. Any macro calls in the definition are

expanded at the time of the definition. The graphics commands themselves are interpreted for a particular point-size at the time of use, on the basis of the ratio between the selected point-size and the design size.

3. Cross-reference to existing definition

In this form the character is defined indirectly, by citing an existing character definition, eg `Times12:'A'`. The character reference may optionally be followed, in parentheses, by arguments *side* and *top*, specifying an additional displacement of the referenced shape.

4. Composition

A character may also be defined as a composition, by presenting a sequence of the simple forms separated by commas. The first simple form in a composite definition may be any of the simple forms; the remainder must be of the cross-reference type.

Short-hand font definition

A special form of font definition allows individual names to be assigned to particular point-sizes of a scaleable font. As well as providing a separate name for the selected size, this definition serves to declare an intent to make significant use of this size. Example:

```
@D.FONT SCRIPT12=SCRIPT(12)
```

No further information is included in this form, and there are no terminating semi-colons.

Complete Examples

The creation of original alphabetic fonts is a specialised art, and their representation and attributes must be worked out with great precision. However, it is relatively straightforward to define fonts derived in a variety of ways from existing fonts, although again care requires requires to be exercised to achieve acceptable effects.

In order to define a new fixed pitch font `MYLIST9` from an existing font `LIST8` with modified horizontal spacing, but otherwise identical:

```
@D.FONT MYLIST9=LIST8 23;
33:126=23;
;
```

In order to define a new font `Smallcaps12` from an existing font `Times12` by replacing all lower case letters with the upper case letters from another existing font `Times8`, the following font definition could be made:

```
@D.FONT Smallcaps12=Times12;
;
'a':'z'=Times8:'A':'Z';
```

Font Selection

`@`*font-identifier*
`@`*font-identifier*(*point-size*)

Once a font has been defined by means of the D.FONT directive, the font identifier is available for use in a font selection control sequence. Any occurrence of the identifier following the escape marker constitutes a selection of the font. In the case of a scaleable font, the required point-size is given in parentheses following the font-name.

Font selection does not imply any kind of break.

Additional forms of font selector are provided by the F directive.

Subscripting and superscripting

A font selector may optionally be followed by a plus or minus sign and a numeric expression. This form is used to specify a vertical displacement to be applied to characters selected from this font. In conformity with the standard interpretation of the vertical co-ordinate, the plus form denotes down (subscript) and the minus form denotes up (superscript).

The numeric expression may denote a measure giving the absolute amount of the displacement, or a number, which is taken as a multiple of the *above-height* plus *below-height* of the font current prior to the new selection. The default value is 1/3.

Example

```
X@times8+[1]+Y@times8+[2] = Z@times8-[3]
```

$X_1+Y_2 = Z^3$

Scope of font selection

The effect of a font selection, including any subscripting or superscripting effect, may be delimited by following it by one of the scope brackets [or {. In that case, the pre-existing font is restored at the end of the scope. If the font selection is not scoped, the effect persists until the font is explicitly changed by another font selection, or restored by the ending of a global scope or block.

Single character scope

When only a single character is required from a particular font, the font selector may be followed by a colon and an integer expression for a character-code, for example, `@TIMES17:'@'` or `@TIMES17:163`. In this case, the selection applies to the one character designated.

Above-height

This specifies the displacement from the top vertical position to the base-line appropriate for characters in this font. In general, it should not be less than the maximum extension of any character in the font above the base-line.

Below-height

This specifies the displacement from the base-line to the top of the next line appropriate for characters in this font, In general, it should not be less than the maximum descender size of any character in the font.

Formatting widths

Following the general font information, comes information about which characters are defined in the font and their formatting widths. This takes the form of a list of width specifications separated by commas and terminated by a semi-colon. Each element in the list consists of a character range followed by an equals-sign followed by an expression defining the formatting width.

A character range is either a single expression denoting a character code (eg 17 or `'A'`), or two such expressions separated by a colon denoting all the character codes between the first code and the second inclusive (eg `'A':'Z'`). Character codes must lie within the limits 0 to 255.

Formatting width is the effective width of a character for horizontal spacing. It is a *measure*, with a default unit of device pixels, and must be strictly positive. One of the ways in which it may be defined is by cross-reference to a character in an existing font, using the form *font-identifier*`:'A'`, for example *TIMES14*`:'A'`. This form can also be used to define the widths of a *range* of characters, using the form *font-identifier:code-value:code-value*, for example *TIMES14*`:'A':'Z'`.

Some examples of the formatting width part:

`0:255=30`	fixed pitch 30 pixels, all characters present
`33:126=25`	fixed pitch 25 pixels, normal Ascii printing set
`33=11, 34=19, ...`	proportionally spaced, individual cases
`33:126=Times12:33:126`	defined with reference to another font

Character definitions

The formatting width section is followed by definitions for the character shapes for all characters to be included in the font, or replaced or added to an existing set.

Each definition is introduced by a character-code or character range (as above), followed by an equals-sign.

In the case of a single character definition, the equals sign may be followed by any of the forms of definition itemised below.

In the case of the definition of a range of characters, the only permitted form is a cross-reference to a range of characters in an existing font, in the same form as in the similar case for formatting widths.

There is no separator between individual character definitions, which should each begin on a separate line. The whole sequence is terminated by a semi-colon.

Define Font

@D.FONT *identifier font-attributes; formatting-widths; character-definitions*

The D.FONT directive introduces a new font and defines the characteristics of the font and the characters contained in it. It provides a means of extending the resident font capability of the printer. A raster-defined font implements a single point-size, while a graphically-defined font implements a range of point-sizes.

The D.FONT is followed by a space and the *font-name*. The information about the font follows in three sections separated by semi-colons.

The font-name may optionally be followed by an equals-sign and the name of a previously defined font, to indicate that the new font is to be derived from the old. In this case the new font will have all the characteristics of the old font, except as modified in the new definition.

Font name

The font-name consists of at least two letters, followed by any number of letters and digits. The name must be distinct from any other identifier defined within the same block. In order to establish the family relationship among raster-defined fonts implementing different point-sizes of the same font family, the font-name of such a font should end with a sequence of digits giving the point-size. For example, defining fonts with the names *MODERN10* and *MODERN12*, establishes the former as the 10-point variant of *MODERN* and the latter as the 12-point variant. However, in these cases the multiple font definitions are otherwise independent of each other. Note that family relationships are established by the actual font-names and hence are not inherited by derived fonts.

If the basic font-name is followed in parentheses by an integer expression, this indicates that the font is scaleable. The characters in such a font should be defined by the graphics variant of image definition, rather than the raster variant. The integer expression gives the point-size corresponding to the direct interpretation of the graphics definitions (the design size for the font), for example *SCRIPT(48)*. In general it is recommended that the design size should not exceed 120 points. Any desired size of a font defined in this way may be selected by an expression such as *SCRIPT(12)* or *SCRIPT(18)*, implying scaling relative to the design size, that is, 12/48 and 18/48 respectively for the examples given. Note that the only name directly defined by this type of definition is the family name (here *SCRIPT*), not variants such as *SCRIPT12*.

Font attributes

Space-width, *Above-height*, *Below-height*;

The first section of the font definition defines the overall characteristics of the font and comprises a list of expressions separated by commas, and terminated by a semi-colon. There are no defaults for these values unless the font is being defined with reference to an existing font, in which case the default is the corresponding value in the other font. The values are *measures*, with a default unit of device pixels.

Space-width

This specifies the horizontal spacing increment to be used for this font as a normal space width.

End Block

@E
@E.*sub-identifier*

The E directive marks the end of a block introduced by the S (Start Block) directive. All definitions made between the corresponding S directive and the current point in the source are discarded. In addition the current formatting state is restored to the formatting state at the point of the S directive.

For the simple E directive, it is an error if the last block was not opened by a simple S directive. However, an extended form of the E directive, in which the E is followed by a period and a block identifier, may be used to terminate a named block. In this case, all blocks, named or anonymous, up to and including the named block are terminated.

The E directive implies a line break.

End Document

@E*
@E*.*sub-identifier*

The E* directive terminates any blocks started within a document, named or anonymous.

The plain E* directive, without following sub-identifier, marks the end of a distinct document.

The E* directive with attached sub-identifier terminates a document, started by an S* directive. It does not imply end of document, to allow for further definitions to be added to a containing document block. It should therefore be followed by a plain E* directive if the intention is to cut back to the containing document block.

The E* directive implies a line break.

Font size

`@F(`*point-size*`)`
`@`*point-size*

The F directive has the effect of selecting an alternative point-size of the currently selected font family. For example, with font *List12* selected the directive `@F14` would select font *List14*. Similarly, having selected *SCRIPT(12)*, the directive `@F18` would select *SCRIPT(18)*. In the case of a non-scaleable font, the nearest point-size is selected if there is no exact match.

The default value for the argument is the current point-size. Used as a basic font selector, this would be a vacuous operation, but it is relevant in association with the presence of following qualifiers.

In the case that the integer expression for the point-size is a simple decimal numeral, rather than a variable or expression, the letter F may be omitted, that is, the escape marker may be immediately followed by the point-size, for example `@12`.

The F directive does not imply any form of break.

Qualifiers

A Font-size directive may be followed by any of the qualifiers which may be applied to a full font-name used as a font selector: superscripting and subscripting, scoping, and single character selection.

For example, the form `@F:163` would denote character number 163 in the current font.

The earlier example of sub-scripting and super-scripting could be more compactly and flexibly expressed in the following way:

```
X@8+[1]+Y@8+[2] = Z@8-[3]
```

$X_1+Y_2 = Z^3$

Declare variable

`@D.VAR` *identifier-list*
`@D.VAR`*(length)* *identifier-list*

The D.VAR directive is used to declare either arithmetic or string variables.

For string variable declarations, the directive is immediately followed in parentheses by an integer expression giving the length of the largest string which the variable may require to contain.

The D.VAR is then followed by a list of variable identifiers separated by commas. Each identifier consists of at least two letters followed by any number of letters or digits. The identifiers must be unique among all identifiers defined for any purpose within the same block.

Any of the identifiers may be followed by an equals-sign and an expression, specifying an initial value for the variable. If no initial value is given, a numeric variable is initialised to zero and a string variable is initialised to the empty string.

Examples

```
@D.VAR Count,Distance,NumberPerLine=20
@D.VAR(80) Culprit, Message='Rejected labels:'
```

Variable invocation

`@`*variable-identifier*

The current content of a variable may be incorporated in the source text stream by including an occurrence of the variable name immediately following the escape marker, like a macro call without arguments.

The content of a string variable invoked in this way is subject to normal processing, including interpretation of control sequences.

The content of a numeric variable invoked in this way is presented in a standardised decimal form. More flexible forms of presentation for numerical variables, and the possibility of incorporating the values of compound expressions, are available through the V (Value) directive.

Variable assignment

`@`*variable-identifier=expression*

This form of control sequence is used to assign a new value to a variable, replacing the existing value. The type of the expression must be appropriate to the type of the variable (arithmetic, string).

On execution, the expression is evaluated and the value is assigned to the variable, overwriting the existing value.

Examples

```
@Distance=3"
@Count=Count+1
@Message=Message.' '.Culprit
```

Define Format Macro

@D.FORMAT *identifier*(*argument-specifiers*) = *macro-body*

The D.FORMAT directive introduces the definition of a special kind of macro, called a *format macro.* Format macros provide the means of defining page formats which can be activated on a continuing basis to provide a standardised layout along with heading and other background information for each page.

The syntax of the definition and call of format macros follows the pattern of conventional macros. The essential difference is that processing of a format macro can be *suspended* by a special form of the O (Open page) directive and then *resumed* when the page opened by that directive is closed.

When a format macro is called initially, processing starts off in the same way as for other macros. The format macro inherits the environment current at the point of call. However, the formatting environment at this time is preserved as the *calling environment.* In the course of processing the format macro, if the special form of the O directive is encountered, processing reverts to the point of call with the new page environment but otherwise the previous state of the calling environment. On suspension, the now current environment is preserved as the state of the format macro.

Processing continues from the point of call. When subsequently the virtual page is closed, whether implicitly as a result of page filling or explicitly through the use of the C (Close page) directive, processing of the format macro is resumed at the point of suspension and with the environment current at suspension. The environment then current in the main line of processing is preserved as the calling environment.

Depending on the particular case, the format macro may create some printing effects. It may then terminate, or open another virtual page perhaps after closing its containing page. Whenever the special form of the O directive is encountered, processing reverts to the point at which main line processing was interrupted. In this way processing alternates between main line text and the format macro, each with its independent formatting environment.

Format macros can call ordinary macros but not other format macros. Hence, a format macro cannot call itself recursively. Instead, a call of a format macro from itself causes the macro to be repeated from the beginning. As a protection against indefinite looping, this repetition is rejected unless the format macro has executed at least one special Open Page directive during the preceding execution cycle.

A call (from outside) of a suspended format macro also has special significance. First the current activation of the format macro is terminated. The format macro is then restarted, unless the format macro identifier is immediately followed by a minus-sign.

Example

This example defines a macro *double* which creates a page format suitable for listing text in a double-A5 format in landscape mode. It contains two Open Page directives which open respectively the left-hand and right-hand sub-pages. It then closes the containing page, which would usually be the physical page, before repeating the complete macro by means of the contained call of *double.*

Both the sub-pages have the same size and orientation (East). They differ only in the position from the top of the containing page, 0.5" for the first and 5.75" for the second.

```
@D.FORMAT double
=@O>(0.5",0.5",7",5")[]
=@O>(0.5",5.75",7",5")[]
=@C
=@double
```


Graphics mode

@G(*scale,Xzero,Yzero*)

The G directive is used to switch into graphics mode. While graphics mode is in force, the source data is interpreted as a sequence of graphics commands. Although the overall environment for the execution of graphics commands is the same as in text mode, there are a number of significant differences between the two. Many of these stem from the fact that in graphics mode, the source data is interpreted as commands, not text.

a. directives are **not** preceded by the escape marker

b. text to be included in the document has to be quoted

c. there is a different set of built-in directives

d. graphical operations are immediate in effect

e. composition effects are restricted to individual sequences of quoted text

f. unquoted spaces and end-of-lines in the source data are not significant

Graphics mode is cancelled by any occurrence of the escape marker (except in a quotation context). This makes it quite straightforward to switch between graphics and text modes to achieve combined effects.

Switching between graphics mode and text mode does not alter the current page attributes or current position (horizontal and vertical). However, switching into graphics mode causes completion of any partially filled line which may be pending, and placing of the line in the page image, without line increment.

Scaling factor

The first argument to the G directive specifies a scaling factor to be applied to all graphical operations. The factor may be fractional. Scaling is applied to all dimensional arguments for the graphical commands, including radii and widths. For example:

`@G3`	Switch to graphics mode, scale factor three
`@G1.5`	Switch to graphics mode, scale factor one and a half
`@G50%`	Switch to graphics mode, scale factor a half

The default scaling factor is unity.

Defining zero position

The arguments *Xzero* and *Yzero* provide a means of specifying a new zero point within the virtual page co-ordinate system, which is treated as a base point for subsequent co-ordinate addressing. The default values for both arguments are zero, which implies no transformation of the co-ordinates appearing in graphics commands.

The value Xzero is added to any subsequent X co-ordinate and the value Yzero is added to any subsequent Y co-ordinate, prior to scaling. Either of the arguments may be negative.

An exclamation mark (!) preceding either of these two arguments implies a change of orientation of the associated axis. That is, the relevant co-ordinate is subtracted from the stated zero value, rather then being added to it.

Example

`@G2,2",!H` Switch into graphics mode with scaling factor of 2. Establish the X zero position 2" from the left of the virtual page and the Y zero position at the bottom of the page (given by pre-defined variable *H*) with Y co-ordinates running up the page. Such a change to the Y zero creates a set of coordinates conforming to conventional mathematical practice.

Co-ordinate units

If no unit is given for any of the dimensional arguments for the graphics commands, they are taken as a number of device pixels. Note the difference, in this respect, between the graphics commands X and Y and the text directives `@X` and `@Y`.

Clipping

Graphical drawing operations are restricted to the current page. Any shapes which would lie wholly or partially outside the page boundary are clipped at the boundary. However, the current position may be set outside the current page boundary, provided that it remains within the representation limits set by the implementation of the protocol.

Text in graphics mode

In order to include character strings in graphics mode, it is necessary to quote them, using the single-quote character (') as a delimiter. The quote character itself can be included by duplicating it. Quoted strings in graphics mode may not extend over a source line boundary, and may not include text mode directives.

Text mode facilities

Font selection, macro calling and other identifier invocations are available in graphics mode by citing the relevant identifier without preceding escape marker. So is assignment to user variables, but not to the built-in variables.

The commands F, Q and V are available in graphics mode with the same significance as the corresponding text mode directives.

Scope brackets in Graphics mode

Enclosing a sequence of graphics commands within scope brackets ([] or {}) causes the sequence to be treated as a closed group. The most basic effect of this is that on termination of the scope, the current position X and Y and the ink widths revert to their values at the start of the scope.

Other uses of bracketed sequences are described under the heading of individual commands such as M (Mark), P (Project) and T (Tile).

Macro Call

`@`*macro-identifier*(*argument-values*)

Once a macro has been defined by means of the D directive, the macro identifier is available for use in a macro call control sequence. Any occurrence of the identifier following the escape marker constitutes a call of the macro. Arguments for the macro on the current occasion of use are placed in parentheses after the identifier.

An argument is either a text string enclosed within single quotes, or an arithmetic expression. In the latter case, the expression is evaluated at the time of the call, and the resulting literal numeric value is taken as the actual argument.

The effect of a macro call is that the body of the macro is processed, with the actual arguments replacing occurrences of the argument identifiers within the body. The text making up the body is interpreted in the environment prevailing at the point of call. There are two exceptions to this:

- The alternative escape marker is recognised at the time of definition, not at the time of use. This avoids possible problems if the alternative escape marker has been changed between definition and use.
- As noted above, argument identifiers in the definition are recognised at the time of definition.

To maintain propriety of structure across macro calls, certain constraints apply to what may be included in a macro.

- Scopes and blocks must be properly nested within macros. That is, a macro may not start a scope or block which it does not end and may not end a scope or block which it does not start. However, a macro does not itself constitute a block, so that macros can include definitions which persist after the macro call.
- Macro and other definitions must be properly contained within macros. That is, if a macro includes the start of a definition, it must include the complete definition.

Macros may include calls to other macros, and recursive calls are also permitted in association with conditional processing (employing the Q directive). The use of recursion provides a means of repeating material on a conditional basis. There is a modest limit on the depth of recursion which is permitted, as a check on excessive or infinite nesting.

Examples (for earlier definitions)

Call	Expansion
`@subpara`	`@B2,88X3`
`@countup`	`@count=count+1` `@q(count=10) @stars=stars.'*' @count=0`
`@descr'Costings'`	`@h[Costings]@X*2.05"`

`@subsection'Current Year'`	`@b1,7`
	`@subsecno=subsecno+1`
	`@v(subsecno): Current Year`
	`@b0`

Define Macro

@D *identifier(argument-specifiers)* = *macro-body*

The occurrence of a D directive introduces a macro definition. The letter D is followed by a space and then the macro identifier - the name to be given to the macro being defined.

For a macro with arguments, the identifier is followed, in parentheses, by a list of *argument specifiers* separated by commas. An argument specifier consists of an identifier optionally followed by an equals-sign and a string expression. An argument identifier consists of at least two letters followed by any number of letters or digits and must be distinct from any other identifier in the same list. If the optional string expression is present, it specifies a default to be used if no argument is supplied for the corresponding argument position when the macro is called.

The macro header (that is, the macro-identifier and argument specification, if any) is followed by the text which constitutes the body of the macro, presented in either of two forms.

- The first form may be used for all macro definitions. In this case the macro header is followed, on the same line, by an equals-sign and then the body of the macro presented as a literal string, following the standard conventions for quoted strings, including line continuation. For example:

```
@D subpara='@B2,8@X3'
```

- Alternatively, when the body of the macro comprises a number of complete lines, the macro header may be followed by a line-end, and then the lines making up the macro body, each preceded by an equals-sign. In this case, the body continues up to the first line not starting with an equals-sign. This form is more convenient for lengthy macro definitions, since it avoids the need to duplicate embedded quote marks. For example:

```
@D countup
=@count=count+1
=@q(count=10) @stars=stars.'*' @count=0
```

Argument identifiers

Within the body of the macro, any occurrence of one of the argument identifiers preceded by the escape marker is treated as a place-holder for the corresponding argument. The recognition of such occurrences is made at the time of the definition and hence takes precedence over any other interpretation of identifiers following escape markers and is insensitive to quotation.

The termination conditions for these occurrences of the argument identifiers following escape markers are the same as for control sequences. Accordingly they should be terminated by a semi-colon, which will be ignored, if none of the other termination conditions applies.

Complete Examples

```
@D DESCR(what) = '@h[@what]@X*2.05"'
@D SUBSECTION(title)
=@b1,7
=@subsecno=subsecno+1
=@v(subsecno): @title
=@b0
```


Graphics positioning commands

These commands establish a new current position. For the X and Y commands, the arguments are interpreted as absolute (that is, zero-point-based). For the R, L, D and U commands, the arguments are interpreted as relative to the current position; the mnemonic significance of these commands fits the standard orientation of co-ordinates, that is, X increasing left to right and Y increasing top to bottom.

X(*Xpos*) **Set X**
set horizontal position to *Xpos* (relative to Xzero)

Y(*Ypos*) **Set Y**
set vertical position to *Ypos* (relative to Yzero)

R(*Xinc*) **Right**
add *Xinc* to horizontal position

L(*Xdec*) **Left**
subtract *Xdec* from horizontal position

D(*Yinc*) **Down**
add *Yinc* to vertical position

U(*Ydec*) **Up**
subtract *Ydec* from vertical position

Set Ink

I(*Right-width,Left-width*)

The I command establishes the ink width to be used in subsequent drawing operations. To allow for flexibility in the way in which the drawing width is distributed on either side of the trajectory of movement, there are in fact two widths: the Right-width and the Left-width. The Right-width determines the drawing width to the right of the trajectory and the Left-width determines the drawing width to the left of the trajectory. The default widths, which are reset every time Graphics mode is entered, are Right-width 1 and Left-width 0.

Instead of a single value, a pair of values separated by a colon may be given for either of the widths. In this case the first value(s) define the starting width for the next line drawing operation and the second value(s) determine the ending width. Varying width applies to straight lines only, not arcs. In the absence of such a specification, drawing operations are carried out with constant ink width.

For certain purposes, where only the trajectory of the line is important, it is legitimate to set both widths to zero.

Graphics drawing commands

The following commands cause either a straight line or an arc or a curve to be drawn **from** the **current** position **to** the **new** position specified in the command. After execution of any of these commands, the new position becomes the current position.

For the M command, the arguments are interpreted as absolute (that is, zero-point-based). For the E, W, S and N commands, the arguments are interpreted as relative to the current position; the mnemonic significance of these commands fits the standard orientation of co-ordinates, that is, X increasing from left to right and Y increasing from top to bottom. The effect of any of the four relative commands can be obtained by each of the others. All four are provided for mnemonic convenience and compactness: by appropriate choice, any case can be covered with non-negative arguments.

The width and positioning of the path drawn are determined by the settings of Right-width and Left-width: the width of the line to the right of the trajectory is given by Right-width, and the width to the left of the trajectory by Left-width.

By default, these commands cause a line to be drawn. However, the prior execution of an A command (see below) causes them to draw an arc instead.

If the M command appears without arguments as the last command in a closed group (sequence enclosed in scope brackets), the endpoint for the drawing executed by this command is the original X and Y current at the start of the sequence.

M(*Xpos,Ypos*) **Mark to X,Y**
draw line/arc to *Xpos,Ypos* (relative to Xzero,Yzero)

E(*Xinc,Yinc*) **East**
draw line/arc to *X+Xinc,Y+Yinc*

W(*Xdec,Ydec*) **West**
draw line/arc to *X-Xdec,Y-Ydec*

S(*Yinc,Xdec*) **South**
draw line/arc to *X-Xdec,Y+Yinc*

N(*Ydec,Xinc*) **North**
draw line/arc to *X+Xinc,Y-Ydec*

Close Page

@C

The C directive marks the end of a page. Its direct effect is to close the current page, which will usually result in a new page being opened to accommodate subsequent text.

The C directive first implies a line break.

If the current page is the physical page, closing it causes the page image to be printed, and a new page to be initialised.

If the current page is a sub-page, opened by the O (Open Page) directive, closing it causes the immediately containing page context to be re-instated. If the page being closed was opened from a format macro, processing of the format macro is resumed.

The effect of closing a page implicitly as a result of page-filling or use of directives such as B or P is the same as when closed explicitly by a C directive.

Closing outer pages

@C.*format-identifier*

A special form of the C directive allows for closing of virtual pages up to a determined level. In this form the C is followed by a dot and an identifier, which must be the name of a defined and suspended format macro. The effect is that pages are closed up to and including the closing of a page within the stated format macro.

Terminating format macros

@C.*format-identifier-*

In the special form of the C directive just described, the format identifier may be followed by a minus (-) to indicate that the format macro is to be terminated once the closing has taken place. In the absence of this qualifier, the format macro continues execution.

- When spacing values are given as a number of lines rather than as a measure, the effective line size is determined in a similar way to the computation of the vertical increment after printing a line, under the control of the Line-increment variable L:

  if the value of *L* is categorical
  or if the value of *L*
  is greater than the current font *above-height* - the value of *L*
  plus the current font *below-height*
  otherwise - the current font *above-height* plus *below-height*

- If the amount of spacing specified exhausts the current page, the current page is terminated. There is no carry over of excess spacing to the following page. As an example, if there is only 3/4 inch left on the current page when a `@B1"` is encountered, the page is terminated and the residue of 1/4 inch is ignored. Note that this applies to a single use of the B directive only, so that, in this respect, the use of two consecutive B directives such as `@B1"@B1"` will not necessarily have the same effect as the single directive `@B2"`.

- Although it is natural to refer to blank space and blank lines being left, it should be noted that all that is involved is a certain vertical movement on the page. No erasing takes place, so that in cases where material has already been printed in the part passed over, it is left undisturbed.

Page Break

The B directive can also be used to achieve a conditional page break, for example to avoid starting a new section of material too close to the end of a page. The page break is taken if, after the line break has been taken and blank space inserted, there is less than a specified amount of room left on the current page for additional lines. The minimum amount of space required is indicated by the second argument following the B. Like the first argument, the second argument may be a measure or a number of lines. For example:

`@B1,0.5"` terminate current line, leave one blank line, and terminate current page if there is less than 1/2 inch of space left on the page.

The default value for the second argument is zero.

Alignment

In most cases where it is required to force a line break, the appropriate treatment for the line being terminated is that it should be left-aligned, without justification (even if justification is in force). The position for left alignment is the left fix-point: that is, the left margin or the latest position established by a horizontal positioning operation such as tabulation. Left alignment is the default treatment for lines terminated by the B directive. However, alternative treatments may be requested by following the B with an alignment code. The possible cases are:

`@B<` Left align (to left fix-point) -- the default
`@B>` Right align (to right margin)
`@B^` Centre (between left fix-point and right margin)
`@B<>` Justify (increase gaps to fill out between left fix-point and right margin). This case is relevant only rarely when line termination is forced: after all, the line may, in the general case, be of any length down to a single word.

The alignment code immediately follows the letter B, and precedes any arguments.

Line-filling mode

The B directive sets line-filling on, that is, it implies `@L1`. In order to leave line-filling off, it should be followed by `@L0`.

Examples of line drawing

{Drawing clockwise with Right-width}

```
i0.06" e2" s0.5" w2" n0.5"
```

{Drawing anti-clockwise with Right-width}

```
i0.06" s0.5" e2" n0.5" w2"
```

{Drawing clockwise with Left-width}

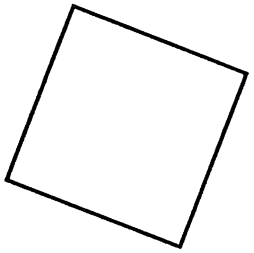
```
i0,0.06" e2" s0.5" w2" n0.5"
```

{Drawing anti-clockwise with Left-width}

```
i0,0.06" s0.5" e2" n0.5" w2"
```

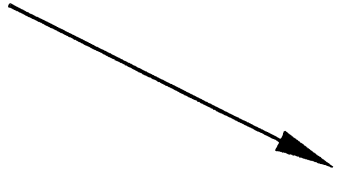
{A tilted square}

```
i8,0 n1.3",0.5" e1.3",0.5" s1.3",0.5" w1.3",0.5"
```



{An arrow}

```
I4,4 S1",-2" I25:1,25:1 S0.2",-0.4"
```



Orb

O(*radius*)

The Orb command O is used to draw solid black circles. The argument specifies the radius of the circle and the orb is centred on the current position, which does not change.

Example

{Varying size orbs}

```
03 R24 05 R24 08 R30 010 R45 015
```


Project

P(*angle,radius*)

The project command P is an alternative to the M command for drawing lines and arcs. It differs in the way in which the endpoint for the drawing operation is specified. For the P command, the endpoint is specified by an *angle* and a *radius*. The radius defines the distance of the endpoint from a *reference point* and the angle defines the direction of drawing with respect to the current position.

The P command can only be invoked within a bracketed sequence of commands. The current position at the start of the sequence is the reference point for projection. The command causes the next line to be drawn from the current X and Y, in the direction determined by *angle*, to the point distant by *radius* from the reference point. The coordinates of the endpoint become the new current X and Y, but the reference point remains unchanged.

The *angle* is specified in degrees.

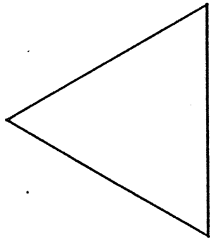
The angle can be given as absolute, with 0 representing due East (increasing X with no change in Y) and ascending values passing through increase of Y then decrease of X i.e. clockwise in the standard orientation.

Alternatively the angle can be given as relative to the previous angle, using the form `P+` for an increase in the angle and `P-` for a decrease.

Example

{A regular triangle}

```
[I0 P60,1"
I4 P+120,1" P+120,1" P+120,1"]
```



5. INDIVIDUAL DIRECTIVES

Blank

@B(*spacing,page-break-threshold*)

The B directive is used to cause a line break and introduce a specified amount of inter-line spacing. It may also be used to cause a conditional page break, if there is less than a specified amount of room left on the current page.

Line Break

The first effect of the B directive is to terminate any partially filled line of document text. That is, it forces a line break in the document at the point where it occurs, over-riding the process of line-filling, even though additional words might fit on the line. It is used at those points in a document where it is inappropriate for the following text to continue on the same line as the preceding text.

`@B0`	terminate current line (if any) with no extra spacing

- The main aspects of the Line Break operation are:
  a. possible termination of the current page **before** the current line as a consequence of page-filling
  b. calculation of vertical position for placing the characters making up the line
  c. horizontal alignment (see below)
  d. calculation of vertical increment to next line position
- The Line Break operation has no effect if there is no partially filled line to be terminated. As a consequence, `@B0` may be freely used to ensure that the following text starts a fresh line, without fear of introducing unwanted empty lines if it should happen that there is no preceding line to be terminated.

Blank spacing

In addition to terminating the current line, the B directive causes a specified amount of extra blank space to be introduced between the previous line and the next one on the page. The amount of space required is indicated by the first argument following the B. For example:

`@B0.25"`	terminate current line and leave 1/4 inch blank space
`@B0.5"`	terminate current line and leave 1/2 inch blank space

In these examples the amount of blank space is given as a measure – in inches. Instead the amount can be given as a number of lines. For example:

`@B1`	terminate current line and leave one blank line
`@B2.5`	terminate current line and leave two and a half blank lines.

The default amount, if no argument is present, is one; that is, one blank line is introduced.

Diagram 4b illustrates the placement of two consecutive lines of text, with thick lines indicating baselines, formatting widths of characters and above and below heights of the fonts used.

A character is not placed in the page image if it would lie wholly outside the dimensions of the current page, for example, as a result of over-long lines of text when automatic line-filling is disabled. However, because glyphs may extend outside the region implied by their formatting width, a character which lies partially outside the dimensions of the current page is placed, provided that it lies wholly within the printable region of the physical page.

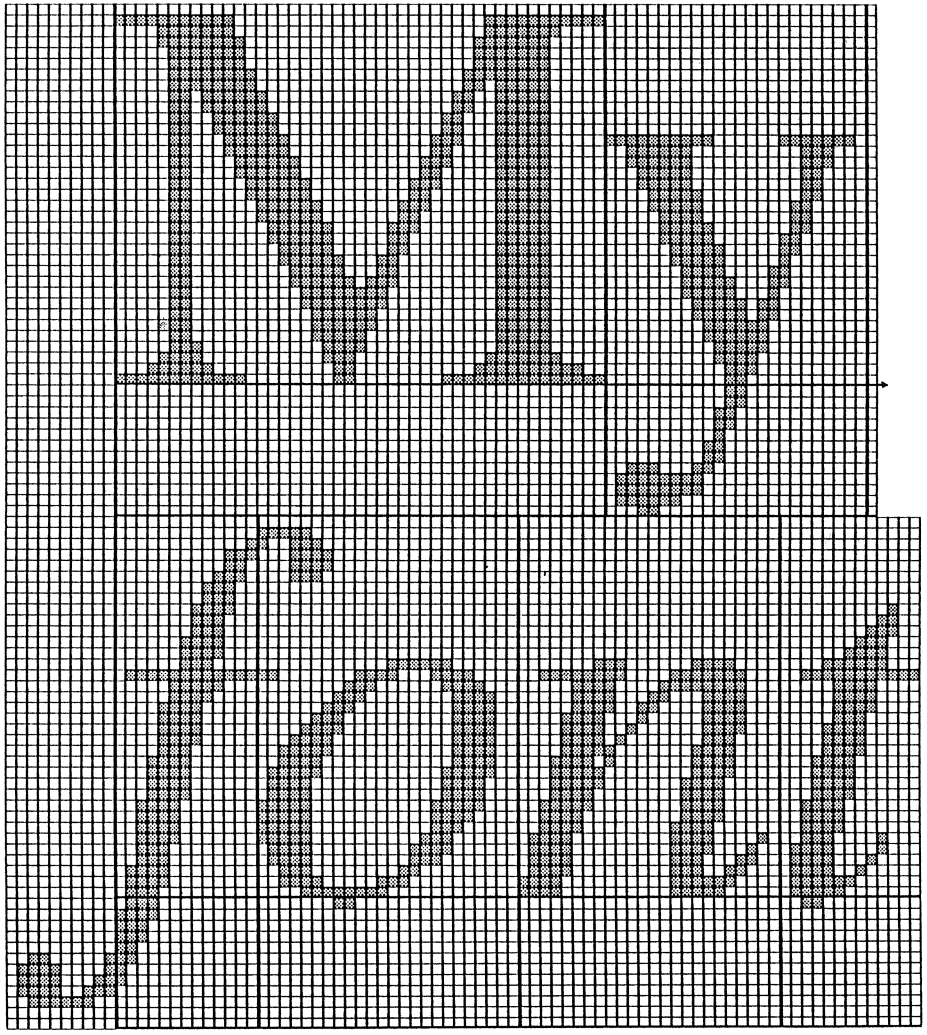


Diagram 4b

Drawing arcs

A(*radius*)

The A command causes the next drawing command to draw an arc of the stated radius, rather than a line. This applies to a single drawing operation only.

If the radius is positive, the centre of the related circle is understood to lie to the right of the trajectory; otherwise the centre of the circle is understood to lie to the left of the trajectory. In either case, the shorter of two arcs is understood. Hence arcs of more than 180 degrees must be drawn in two operations.

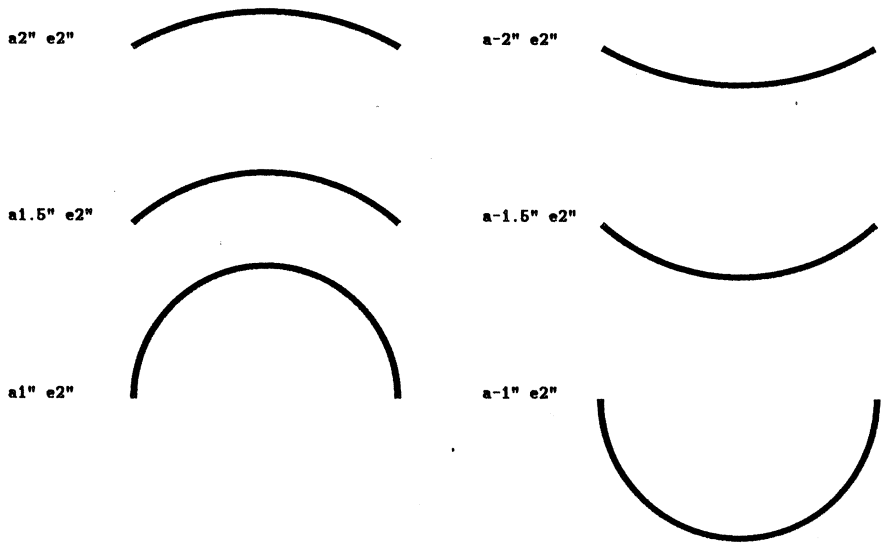
Examples of arc drawing (right-width 0.05 inch)

{Drawing arcs with centre on right}

```
a2" e2"
a1.5" e2"
a1" e2"
```

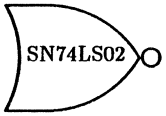
{Drawing arcs with centre on left}

```
a-2" e2"
a-1.5" e2"
a-1" e2"
```



{A NOR logic gate}

```
i6e0.3"a0.8"e0.693",0.4"a0.075"e0.15"
a0.075"w0.15"a0.8"w0.693",-0.4"w0.3"i0,6a-0.8"n0.8"
r0.15"d0.30" 'SN74LS02'
```



Placing bitmap images

B(*width,height,side,top*)

In addition to the line drawing facilities, provision is made for the direct inclusion of fully specified raster images in bitmap form. Such images are introduced by the B command. Following the letter B the form is the same as the bitmap variant of font character definition (see the description of the D.FONT directive).

The bitmap is positioned with the top left corner aligned to the current position.

Bitmaps are not scaled.

Tiling closed outlines

T[.....]
T(*tile-designator*)[.....]

The Tile command T provides a method of filling in closed outlines either with solid black or with a repetitive pattern. To fill an area with solid black, the T has no argument; to fill with a selected pattern, the T is followed, in parentheses, by a *tile-designator* which defines the pattern to be used to tile the area.

The *tile-designator* takes the form of a *font-name* followed by a colon and an integer expression for a character-code-value (for example *LIST10*:'+'). The character must be defined as a basic raster or graphics glyph (not by cross-reference or composition).

In either case, the command is immediately followed by an opening scope bracket. The commands within the scope brackets define the *outline* which is to be filled.

Closed outline

The drawing operations within the scope brackets must define a single concave closed outline, that is, the internal angles must all be less than 180 degrees.

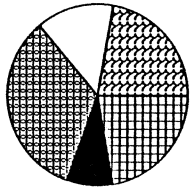
Pattern alignment for filling

The rectangular pattern denoted by the tile-designator may be thought of as replicated across the current page on a fixed grid basis starting from the top left corner, like *tiles*. However, only the region of the page defined by the closed outline is filled with the pattern. This approach ensures matching of patterns when abutted outlines are filled with the same tile.

Example

```
{Tiling of pie-chart slices:-
   in each group the first P draws a line
   from the centre outwards; the second P
   draws the arc; and the M closes the outline}

i2,2
t(list10:'+')[p+0,200a200p+80,200m]
t[p+0,200a200p+30,200m]
t(list10:'*')[p+0,200a200p+120,200m]
[p+0,200a200p+50,200m]
t(list10:'?')[p+0,200a200p+80,200m]
```



4.6. Vertical increment

After a line of text has been placed in the page image, the current vertical position Y is updated by an amount controlled by the Line-increment variable L, in conjunction with the font height values. The value of the Line-increment variable can be set by a control sequence in the protocol. The value can be defined to be *categorical* or *provisional*, depending on the form of the assignment. If the value of this variable is provisional, it may be over-ridden by the values *above-height* and *below-height*.

The computation of the vertical increment is as follows:

if the value of the variable *L* is categorical or if the value of *L* is greater than *above-height* plus *below-height*	- the value of *L*
otherwise	- *above-height* plus *below-height*.

The default value for the Line-increment variable L is *provisional zero*, allowing the vertical increment to be determined automatically by *above-height* and *below-height*.

4.7. Mixed fonts and Vertical Placement

In the general case, there may be characters from more than one font in the same line of text. There may also be modification of the vertical displacement within the line (typically to give effect to subscripting and superscripting). Accordingly, the *above-height* and *below-height* values used in determining base-line displacement and line increment are taken as the maximum of the above-height and below-height values of any of the fonts instantiated on the line, adjusted for any variation of the vertical displacement.

4.8. Raster images

The ultimate representation of a glyph on a raster printing device is as a character *raster image*. Depending on the nature of the definition, the raster image may be represented directly or it may be created dynamically.

A raster image is an internal representation of a printable shape in the form of black and white dots. The raster image is a rectangle of the size of the minimum bounding box required to enclose all the black dots making up the shape. The rectangle has a fixed *width* and *height*, both of which must be strictly positive values.

In addition, as part of a character definition, there are two relative positioning values which are used in placing the shape, relative to the current position when the glyph is invoked. These are termed *side* and *top*. The side value indicates a displacement of the image to the right of the placement position. The top value indicates a displacement of the image downwards from the top line determined by the font above-height.

4.9. Character placing and clipping

The placing of a character (that is, of a glyph raster image) in the page image depends on the values of the state variables X and Y, and the base-line Displacement D described above, together with the font *above-height* and the displacement values associated with the individual character glyph *side* and *top*.

The glyph is placed so that it extends to the right of and down from the co-ordinate position given by (*X+side,Y+D-above+top*).

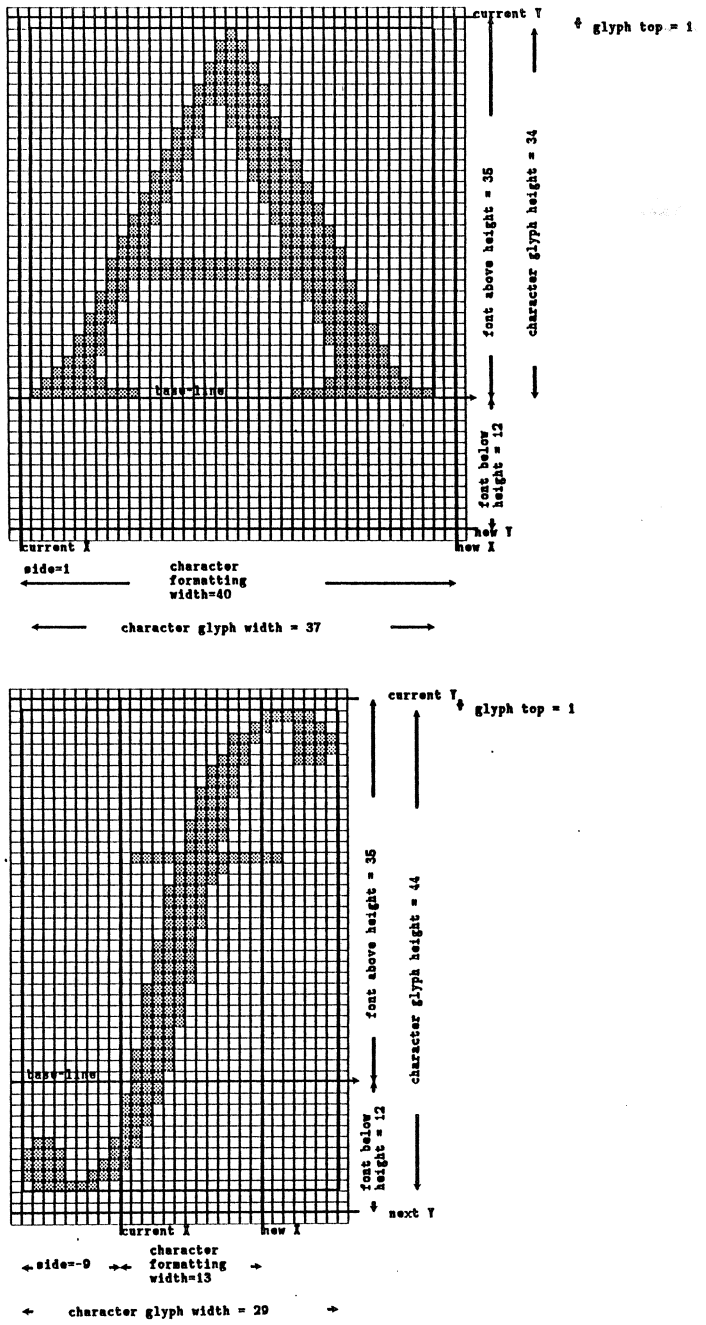


Diagram 4a

Zeroing closed outlines

Z[.....]
Z(*tile-designator*)[.....]

The Z command is used to Zero or erase areas defined by closed outlines defined in the same way as for the T command. It may optionally have a *tile-designator* as argument specifying a pattern to be used to tile the area after erasing.

Complementing closed outlines

C[.....]
C(*tile-designator*)[.....]

The C command is used to Complement (form a negative image of) areas defined by closed outlines defined in the same way as for the T command. It may optionally have a *tile-designator* as argument specifying a pattern to be used to tile the area after complementing.

Highlight

`@H`(*bold-factor*)

The H directive is to used to cause parts of the text to be highlighted, by replicating the character patterns with a small displacement. This is an algorithmic effect, which may be applied to any font. Highlighting implies an increase in the formatting widths of the characters to which it is applied.

The argument for the H directive controls the amount of highlighting as a ratio of the norm defined for each font. The default argument for H is 1; arguments less than 1 imply reduction, and more than 1 increase, of highlighting relative to the norm.

The effect of the H directive may be delimited by following it by one of the left scope brackets [or {. In that case, the pre-existing highlight-factor is restored at the end of the scope. If the H directive is not scoped, the effect persists until the highlight-factor is explicitly changed, or restored by the ending of a global scope, or block.

The H directive does not imply any kind of break.

Examples:

	Text without highlighting
`@H0.5`	**Text with 1/2 of default highlighting**
`@H`	**Text with default highlighting**
`@H2`	**Text with twice default highlighting**

Incline

`@I`(*incline-factor*)

The I directive is used to cause subsequent text to be presented in an inclined or semi-italic form. This is an algorithmic effect, which may be applied to any font. It does not imply any increase in the formatting widths of the characters to which it is applied.

The argument for the I directive controls the amount of inclination as a ratio of the slope of 25% adopted as the norm. The default argument for I is 1; arguments less than 1 imply reduction and more than 1 increase of slope relative to the norm.

The effect of the I directive may be delimited by following it by one of the left scope brackets [or {. In that case, the pre-existing incline-factor is restored at the end of the scope. If the I directive is not scoped, the effect persists until the incline-factor is explicitly changed, or restored by the ending of a global scope, or block.

The I directive does not imply any kind of break.

Examples:

	Text without inclination
`@I1/2`	*Text with 1/2 of default inclination*
`@I3/4`	*Text with 3/4 of default inclination*
`@I`	*Text with default inclination*
`@I3/2`	*Text with 3/2 of default inclination*
`@I2`	*Text with 2 times the default inclination*
`@I4`	*Text with 4 times the default inclination*

In detail, the vertical displacement to the base-line is:

if the value of the variable *D* is categorical	
or if the value of *D* is greater than the *above-height*	**- the value of *D***
otherwise	**- the *above-height***

The default value for the variable D is *provisional zero*, allowing the base-line to be determined automatically by *above-height*.

In LAYOUT a font may have a maximum of 256 glyphs, numbered 0 to 255. Glyphs corresponding to 0 to 32 in the enumeration cannot be selected directly by characters in the source, as these values correspond to ASCII codes for control characters and space. Similarly it may not be possible to access directly by source characters glyphs associated with index values of 127 and upwards, depending on communications options. Alternative mechanisms are provided in the protocol for accessing these character shapes.

Each font also defines a *space-width*. It should be noted that the treatment of the space character in the protocol is not in any way connected with the glyph appearing at position 32 in a font (if any).

4.4. Horizontal increment

Each character in a font has an individual *formatting width* which is defined as part of the font definition and is separate from the character shape information.

The computation of the horizontal spacing increment when a character has been placed in a line is controlled by the Column-size state variable C in conjunction with the character formatting width. The value of the Column-size variable can be set by a control sequence in the protocol. The value can be defined to be *categorical* or *provisional*, depending on the form of assignment. If the value of this variable is provisional, it may be over-ridden by the formatting width.

In detail, the horizontal increment after placing a character is:

If the value of the variable *C* is categorical or if the value of *C* is greater than the *formatting width*	- the value of *C*
otherwise	- the *formatting width*

The default value for the variable C is *provisional zero*, allowing the horizontal increment to be determined automatically by formatting width.

The computation of the horizontal increment for a space is similar to that for characters, using the *space-width* defined for the font in the role of formatting width.

4.5. Base-line

Each font has a defined *above-height* and *below-height*. These values are properties of the font, not of individual characters. They typically represent the maximum extension of any character in the font above or below the *base-line* respectively. The base-line is the point of vertical alignment for characters making up a text line: characters with descenders extend below the base-line; those without descenders do not, or not by much.

The current vertical position Y does not identify the base-line, but the top of the region in which characters are placed.

The computation of the displacement of the base-line down from the current vertical position Y for each line of text is controlled by the base-line Displacement variable D, in conjunction with the font *above-height*. The value of the Displacement variable can be set by a control sequence in the protocol. The value can be defined to be *categorical* or *provisional*, depending on the form of the assignment. If the value of this variable is provisional, it may be over-ridden by the value *above-height*.

Justify

`@J`(*state*)

The J directive is used to specify whether or not justification is to be performed. The possible states are:

`@J0`	justification disabled
`@J1` or `@J`	justification enabled

The state of this option affects only lines which are automatically filled. Justification can also be explicitly requested for individual lines by use of the aprropriate qualifier for the B or L directives.

The J directive first implies a line break.

Character

`@K`(*character-value*)

The K directive is used to include a designated character from the currently selected font. The character code is specified by the argument following the directive which must denote a number in the range 0 to 255 inclusive. The main use of the directive is to access characters with code values outside the standard ASCII printing range. It is also relevant in cases where the character code is not fixed, but is computed as the value of an expression.

The K directive does not imply any kind of break.

Examples

`@K5`	character from current font with code value 5
`@K(flag)`	character from current font determined by current value of variable *flag*
`@K(item-1+'a')`	'a', 'b', 'c', ... according as variable *item* = 1,2,3, ...

Line-fill

@L(*state*)

The L directive determines whether automatic line-filling is to take place or not. The possible states are:

@L0	line-filling disabled
@L1 or @L	line-filling enabled

When line-filling is disabled, the line structure of the printed document follows that of the source. However, composing operations are available within a line, and page-filling is performed. In this mode, each space is significant; a line-end in the source implies a line break in the document; a blank line in the source implies blank line in the document.

When line-filling is enabled, running text is automatically split up into lines, on the basis of how many words will fit within the available page width. Spaces, including multiple spaces, and line-ends in the source serve to mark word boundaries.

The state of line-filling may be freely changed in the course of a document, to allow running text to be filled but leave section headings and itemised lists as they are, for example.

In addition to explicit use of the L directive, line-filling is implicitly enabled by either of the line-breaking directives B (Blank) or P (Paragraph). These directives very commonly appear after the kinds of construction for which line-filling is disabled, so that the need to include L directives to re-enable line-filling explicitly is minimised.

The L directive first implies a line break.

Alignment

As well as determining the state of line-filling, the L directive may be used to specify the alignment required for subsequent lines. This applies only when composing is enabled. The possible cases are:

@L<	Left align lines (to left fix-point)
@L>	Right align lines (to right fix-point)
@L^	Centre lines (between left fix-point and right fix-point)
@L<>	Justify lines (increase gaps to fill out between left fix-point and right fix-point).

The alignment code immediately follows the letter L, and precedes the argument.

If no alignment is specified, then lines which are not automatically line-filled are left-aligned; lines which are line-filled are treated according to the setting of the Justification option.

4. FONTS

At any point in a document a number of *fonts* are available for use. Some fonts are pre-defined, either by being part of the fixed repertoire of the printer or by being created as part of the initialisation of the device. The number and identity of these pre-defined fonts are installation dependent.

At any point in a document there is a single font which is *selected*. The other defined fonts are available for selection. A font is selected by a simple control sequence citing the font name. When a font is selected, subsequent text characters are treated as belonging to that font.

4.1. Font families

A *font family* is a group of fonts designed for use together with each other, each member being an example of the same general style of type design. When the distinction between individual members of the family and the family as a whole is not important, the word *font* by itself is often used to indicate what is here termed a font family.

The only principle of grouping which is relevant for LAYOUT is the attribute of size. The way in which fonts are defined can establish that a group of font definitions, or a single scaleable font definition, provide a variety of sizes of the same font family. There are control sequences which can be used to select an alternative size of the currently selected font. The font selected by such a control sequence is the appropriate member of the font family to which the current font belongs, whatever that may be.

4.2. Font names

Fonts are named by identifiers constructed according to the pattern: *family-name* followed by optional numeric *point-size*. Some examples are *TIMES10*, *TIMES12*, and *HELVETICA24*. In the last example, the family name is *HELVETICA* and the point-size is *24*.

The function of the *point-size* in the font name is to indicate the size of the font in terms of the customary printing unit of the *point* (taken as 1/72 of an inch). Its function is to establish the relative size of different fonts; it does not directly determine the size of individual characters in the font.

In the case of a scaleable font, there is a single basic font name without attached point-size, for example *SCRIPT*. Any size of such a font may be selected by including the desired point-size in parentheses as an argument to the family name, for example *SCRIPT(12)* or *SCRIPT(32)*. For convenience and efficiency, there is a form of definition which can be employed to make available frequently required sizes as derived fonts, for example providing the name *SCRIPT12* as equivalent to the expression *SCRIPT(12)*.

4.3. Nature of fonts

An individual font at a given point-size is an ordered set of character representations or *glyphs*, together with associated formatting width information. A text character appearing in the document selects that glyph in the current font which is determined by its character code.

Margins

@M(*left*,*top*,*across*,*down*)

The M directive is used to define margins for the physical page, typically at the start of a document. It can also be used to specify the page orientation.

The first argument *left* specifies the distance of the leftmost part of the new page from the left of the paper.

The second argument *top* specifies the distance of the topmost part of the new page from the top of the paper.

The third argument *across* specifies the width of the page. Alternatively, if negative or zero, it is interpreted as the distance of the rightmost part of the page back from the right of the paper.

The fourth argument *down* specifies the height of the page. Alternatively, if negative or zero, it is intepreted as the distance of the bottom-most part of the page up from the bottom of the paper.

The default values for these arguments are installation dependent. The minimum values for the physical page margins are device dependent, and if smaller values are specified, these are over-ridden by the minimum acceptable values.

The orientation of the page can be specified by an orientation code attached to the M. The four basic orientations are:

@M/	North - the default
@M>	East
@M\/	South
@M<	West

The M directive is valid only when the current page is the physical page. In addition to setting the margins, it also causes the current position to be set to the top left position of the newly dimensioned page. It is further defined to cause a line break, though this is not normally relevant.

The effect of an M directive persists until another M directive or the end of the document.

Open Page

`@O(`*left,top,across,down*`)`

The O directive causes a sub-page to be opened within the current page. The position, dimensions and orientation of the new page are specified. It first implies a line break, completing any partial line in the current page.

The first argument *left* specifies the distance of the leftmost part of the new page from the left of the current page.

The second argument *top* specifies the distance of the topmost part of the new page from the top of the current page.

The third argument *across* specifies the size of the new page in the current horizontal axis. Alternatively, if negative or zero, it is intepreted as the distance of the rightmost part of the new page back from the right of the current page.

The fourth argument *down* specifies the size of the new page in the current vertical axis. Alternatively, if negative or zero, it is intepreted as the distance of the bottom-most part of the new page up from the bottom of the current page.

The default value for all arguments is zero.

The orientation of the new page relative to that of the current page is specified by an orientation code attached to the `@O`. The four basic orientations are:

`@O/\`	North - the default
`@O>`	East
`@O\/`	South
`@O<`	West

Within a format macro, or a macro called from a format macro, a special variant of the O directive is valid. This takes the form of appending an empty pair of scope brackets ([] or {}) to the control sequence. The empty scope brackets symbolise that text to fill the page is to be taken from elsewhere. When this form is used, processing of the format macro is suspended, and processing reverts to the calling environment, with the new page context.

Each Open Page directive creates a context for a single page and its effect ceases when that page is closed; that is, the Open Page directive does not establish a continuing page context. The effect of a continuing page context is achieved by the repeated execution of Open Page directives within a format macro.

When line-filling is **disabled**, each individual space character in the source is significant and each line-end in the source causes a line break in the printed document.

3.13. Measures

Any simple numeric expression, such as a numeral or a variable, or any parenthesised numeric expression, may be followed by the double quote symbol (") signifying inches. Such an expression denotes a *measure* rather than a *number*.

Many of the directives require measures as arguments. In every case a number is also acceptable and there is a stated default measure which is taken as the *unit* for such a number. That is, the number is automatically converted to a measure by multiplication by the unit.

Common units are *columns* for horizontal measures and *lines* for vertical measures. Only in graphics mode are device pixels the default units. The way in which column size and line size are determined is described in the following chapter, and in the description of the B (Blank) directive.

The inch is taken as the basic unit of measurement because of its pervasive usage in typography, but the effect of alternative units can be obtained by multiplying by a variable defined (as a fraction of an inch) to correspond to another unit, say, the millimetre.

A single character literal string may be used in a context requiring an integer numeric expression, in which case it denotes the Ascii code value of the character.

The only string operator is:

concatenation . (dot) eg `hold=hold.' '.extra`

3.10. Words

For composition operations such as justification and line-filling, it is necessary to distinguish groups of characters which make up individual words. Within the protocol, there is a simple definition of a word, namely, any sequence of characters delimited fore and aft by one of the following:

a space
a line-end
a directive defined to cause a word or line break

According to this definition, punctuation symbols count as part of the words to which they are attached.

3.11. Sentences

A sentence is defined to be a sequence of words, the first starting with a capital letter and the last ending with any of the following punctuation symbols:

period (.) question-mark (?) exclamation-mark (!)

The only relevance of the sentence as a unit in the protocol is that extra space is inserted at sentence boundaries, under the control of the G (Sentence-Gap) state variable.

3.12. Spaces and line-ends

The treatment of space and line-end characters within argument lists has been described above. Outside control sequences, spaces and line-ends are significant components of the document text.

When line-filling is **enabled**, running text is composed into lines made up of as many words as will fit in the page width. Both spaces and line-ends in the source serve simply to terminate words. Multiple spaces are treated as equivalent to a single space, and a line-end is treated as equivalent to a space, so that it does not imply a line break in the printed document. In general, in this mode, additional spacing effects are achieved by the inclusion of control sequences. However, certain commonly occurring requirements are covered by treating spaces and end-lines as significant in the following contexts:

a line-end followed by a space or line-end
a line-end preceded by a line-end
spaces at the start of a line

Accordingly, starting a source line with one or more spaces will cause a line break in the document, and the initial spacing will be preserved. Including a blank line in the source will also cause a line break, and the blank line will be preserved in the document.

Among the built-in directives, the majority are defined to have an effect which includes causing a line break – `@B` and `@P`, for example. Others have an effect which includes causing a word break – `@T` for example. The remainder of the built-in directives – `@U` for example – imply neither a line break nor a word break. Similarly, the occurrence of a user-defined macro does not of itself imply any kind of break, though directives in its definition may do so.

Paragraph

`@P`(*spacing,page-break-threshold*)

The P directive is used to start a new paragraph. It is similar in effect to the B directive, but the default values for the arguments are different, and it has the additional property of causing the start of the next line of text to be indented. The description of the B directive contains a fuller description of the effects of Line Break, Blank Spacing and Page Break.

Line Break

The first effect of the P directive is to terminate the current line of text if there is one. That is, it forces a line break in the document at the point where it occurs, over-riding the process of line-filling.

Blank Spacing

In addition to terminating the current line, the P directive causes extra blank space to be introduced between the previous line and the next one on the page. The amount of space required is indicated by the first argument following the P. The value may be a measure (in inches) or a number of lines. The default value for spacing for the P directive is a single line.

Page Break

The P directive can also be used to cause a conditional page break. This facility can be used to avoid starting a new paragraph too close to the end of a page. the line break has been taken and blank space inserted, there is less than a specified amount of room left on the current page for additional lines. The minimum amount of space required is indicated by the second argument following the P. Like the first argument, the second argument may be a measure or a number of lines. The default value for this argument for the P directive is two lines.

Indentation

Finally the P directive causes the immediately following text to be indented. The amount of the indentation is not specified as an argument, but is the value of the pre-defined Indent-paragraph variable *I*. Paragraph indentation applies to a single line only.

Complete examples

`@P`	terminate current line, leave 1 blank line, take page break if remaining space on page is less than 2 lines, and indent following text by *I*.
`@PO`	terminate current line, leave no extra vertical spacing, take page break if remaining space on page is less than 2 lines, and indent following text by *I*.
`@PO.2",1"`	terminate current line, leave 1/5 inch vertical spacing, take page break if remaining space on page is less than 1 inch, and indent following text by *I*.

Query

&Q(*condition*)

The Q directive provides a means of selective processing of parts of the source text according to the result of a conditional test.

The argument to the Q, which must be an arithmetic expression, is evaluated. If the value is greater than zero, the condition is taken to be true. In this case, processing proceeds normally.

If the value is less than or equal to zero, the condition is taken to be false, and processing is suspended.

The scope of conditional processing may be indicated by the use of the scope brackets [ ] or { }. The scope must not include a line-end.

Alternatively, if the Q control sequence is not terminated by a left scope bracket, the scope is the rest of the current source line including the line-end.

When it is necessary to make processing of a number of lines conditional, the same condition can be repeated in front of each, or the group of lines can be defined as a macro and the macro call made conditional.

The default value for the argument for the Q command is zero (false). Consequently the simple directive by itself (plain **&Q**) can be used to cause the rest of the current source line, including the line-end, to be ignored. A special case of this is **&Q** at the very end of a line which has the effect of causing the line-end to be ignored - useful when source lines would otherwise become over-long.

3.8. Arithmetic expressions

Expressions denoting numeric values may be simple terms or compound expressions involving arithmetic operators.

The forms of simple term are:

decimal integer	eg	`0 1 999`
decimal fraction	eg	`0.75 12.50`
variable	eg	`x t(2) pageno`
measure	eg	`1" 2.5" count"`

The operators are:

negation	- (prefix)
addition	+
subtraction	-
multiplication	*
division	/
percentage	% (postfix)
logical and	&
logical or	!
equals	=
less than	<
greater than	>
not-equals	<>
less or equal	<=
greater or equal	>=

The precedence of the operators is:

1 (most binding)	* / % &
2	+ - !
3	= < > <> <= >=

Parentheses may be used in the usual way to group operands, in order to over-ride operator precedence or for clarity.

The relational operators such as 'equals' yield the numeric value one if the relation holds, and zero if it does not.

3.9. String expressions

The two forms of simple string terms are:

literal string	eg	`'A simple text term'`
string variable	eg	`notebuffer`

Literal strings are sections of text enclosed between occurrences of the single quote character (' - Ascii 39_{10}). All characters within the quotation marks are significant, including spaces, line-ends and control sequences. Within a quoted string, control sequences are not subject to interpretation, except in the special case of argument identifiers within macro bodies. The quote character itself can be included by duplicating it ('').

As a protection against runaway errors, when a string continues over more than one line (that is, contains line-ends), it is necessary to include a hyphen continuation character (-) immediately before each such line-end character. The continuation character does not form part of the string. A line-end character occurring within a string and not preceded by a hyphen is treated as an erroneous completion of the string.

An individual argument is either an arithmetic expression or a string expression. Omitting an argument implies acceptance of any default argument defined for this argument position. When an omitted argument is followed by an included argument, the separating comma(s) must be present; but when trailing arguments are omitted, the following commas should also be omitted.

For compactness, parentheses enclosing arguments may be omitted where no ambiguity could arise from so doing. There is the important difference when parentheses are omitted that non-significant spaces and line-ends are permitted only after a comma. A space or line-end in any other position serves to terminate the control sequence.

3.6. Termination of control sequences

If no argument is present, or parentheses are not used to enclose arguments, a complete control sequence requires to be terminated by one of the following:

a space
a line-end or other control character
a letter not following an identifier
a left bracket ([or {)
a right bracket (] or })
an escape marker
a semi-colon, which is ignored

The last form of termination (semi-colon) is provided for the fairly rare cases where none of the others applies, although it may be used in any case. As noted, in this case the terminating character is ignored.

When arguments are enclosed in parentheses, the right parenthesis serves to terminate the sequence. A semi-colon may not be used in this case.

3.7. Scopes

When a control sequence is terminated by either of the left brackets [and {, this has the effect of *scoping* the effect of the directive or assignment. The scope continues up to the next matching right bracket] or }. At the end of the scope, the prior state of the formatting option or variable is restored.

Note that a right bracket is not recognised as a scope terminator unless the last scope was opened by the corresponding left bracket.

A group of control sequences may be scoped simultaneously if they are run together without intervening spaces or other separators.

The scope brackets themselves do not imply any kind of formatting break. However when a directive which itself implies a break is scoped, ending the scope is equivalent to another occurrence of that directive and accordingly implies the same type of break. Relevant cases are the L (Line-fill), X (X-set), and Y (Y-set) directives.

Certain directives are not valid within a scope, and the appearance of any of them forces termination of any open scopes. These are the Define directive (all forms) and the Start and End Block directives.

Right-move

&R(*amount*)

The R directive has the effect of introducing horizontal spacing within text. It provides a means of achieving precise control over the positioning of characters on a line.

The argument specifies the amount of right-movement required. This may be given as a measure or as a number. The latter is interpreted as a multiple of the built-in variable U. The initial value for this variable is the width of the device pixel, typically 1/300", but it may be set to an alternative value by means of an assignment to U.

The R directive does not imply any kind of break. Note the difference in this respect, and in the interpretation of the argument, between the R directive and the X directive.

Control characters for Right-move

For economy of representation of frequently required Right-move operations in low-level use of LAYOUT, four ASCII control characters have a pre-defined significance. These are the Separator controls FS, GS, RS and US (Ascii 28_{10} - 31_{10}), grouped as two pairs: FS with GS and RS with US. They allow the compact definition and use of two independent fixed horizontal movements.

The first character in each pair defines and invokes a Right-move; the second invokes the move last defined by its partner.

For the first character in the pair, the amount of the move is defined by the code value of the single immediately following character, with the code '0' (Ascii 48_{10}) corresponding to zero. The unit of movement is given by the built-in variable U, as for the R directive.

Start block

`@S`
`@S.`*subidentifier*

The S directive causes a new block to be started. A block is a self-contained unit for definitions, and changes of formatting state. The block extends from the occurrence of the S directive to the corresponding E (End block) directive. Any definitions made within the block are local to it and are automatically discarded at the end of the block.

In addition, at the end of the block, the current formatting state is restored to the formatting state at the S directive.

A suffix may optionally be attached to the letter S, to provide an identifier for the block, for example S.MAIN. This identifier may be cited in an E directive to cause all blocks nested within the named one to be ended along with the named one.

The S directive implies a line break.

Start document block

`@S*.`*subidentifier*

A special form of the S directive can be used to start a document block. This takes the form S* and must be accompanied by a sub-identifier to name the block.

Any definitions made globally to a document block are preserved even when a document is ended. Document blocks are ended only by a special form of the E (End block) directive, of the form E* with the relevant sub-identifier. Document blocks cannot be terminated by other E directives.

The plain End Document directive E* does not terminate a document block either, though it cancels all definitions made within it, and restores the formatting state to that active at the start of the document block.

The S* directive is not valid within a macro call or when a format macro is active.

The S* directive implies a line break.

3.4. Control sequences

The different cases depend on the character or characters immediately following the escape marker. When control sequences include letters, it is immaterial whether upper-case or lower-case is used.

(a) *a single letter with optional sub-identifier*

This is the form for a built-in *directive* which has a pre-defined significance in the protocol. If the directive is defined to take arguments, the directive identifier may be followed by one or more expressions giving the desired values of these arguments for the directive on this occasion of use. Directive arguments may be enclosed in parentheses, but need not be in the (typical) case where the argument is a numeral. So, for example, the directive `@B(2)` may be abbreviated to `@B2`. For some directives, there are additional qualifiers which may appear between the directive identifier and any arguments. Examples are the characters plus (+) and minus (-) used as relative qualifiers. Full information about the form of the built-in directives and their arguments is given in the individual descriptions in Chapter 5.

If the identifier is followed by an equals sign and an expression (eg `@I-3`), the control sequence is an assignment, rather than a directive. In this case the identifier denotes one of the pre-defined state variables and the effect of the assignment is to change its value to the value of the given expression.

(b) *two letters followed by any number of letters or digits*

This is the form for a user-defined control sequence. The sequence of characters following the escape marker is an *identifier* for one of a number of kinds of defined objects. The kind of the object depends on the type of the definition.

If the identifier is not followed by an equals-sign, the control sequence is an invocation of the object denoted. The main kinds of object which can be invoked are: fonts, macros, and bitmaps. In the case of a font, this form of control sequence is a *font selection*. In the case of a macro, it is a *macro call*. If the object has been defined to have arguments, the identifier may be followed by the specification of their values on this occasion, within parentheses when necessary.

If the identifier is followed by an equals-sign, the control sequence is an assignment. The identifier must be the name of a previously declared variable and the equals sign must be followed by an expression of the appropriate type for the type of the variable. The effect of executing the assignment is to replace the current value of the variable with the value of the given expression.

(c) *a number (sequence of digits)*

This form is used to specify a font change, the number giving the point-size of the required new size within the current font family. There are various qualifiers which may be appended to the number, for example, to vary the vertical displacement for purposes of super-scripting and sub-scripting.

3.5. Syntax of argument lists

An argument list following an identifier is a sequence of arguments enclosed in parentheses and separated by commas. Within parentheses, spaces and line-ends are permitted and ignored in any of the following positions:

before the left parenthesis
before or after a comma
before the right parenthesis

3.3. Escape marker

A basic characteristic of the protocol is that data is interpreted as text to be printed, unless explicitly flagged otherwise. The control information is embedded in the data stream in the form of *control sequences*, which are not printed in the final document but are interpreted as part of the protocol.

All control sequences are introduced by a special character called the *escape marker*. The control character ESC (Ascii 27_{10}) can always be used for this purpose and the significance of this character cannot be altered. In addition, a single *printing* character can be defined as an alternative escape marker. This makes it straightforward to prepare protocol files using standard utility software such as context and screen editors. The default character used for this purpose can be selected as an installation option. Characters which have a defined significance within control sequences in the protocol are not eligible. The characters which are eligible are:

- `$` - Ascii 36_{10}: dollar-sign
- `@` - Ascii 64_{10}: at-sign
- `` ` `` - Ascii 96_{10}: grave accent
- `~` - Ascii 126_{10}: tilde

Within this document, the generic symbol `@` is used to represent the escape marker.

The alternative escape marker, if defined, is the only fully reserved printing character pre-supposed by the protocol. That is, it is the only printing character which does not stand for itself in running text. However, certain other characters have a reserved significance within control sequences.

The escape marker introduces a control sequence when it is immediately followed by a letter or digit.

When the escape marker is not followed by a letter or digit, its significance is as follows:

(a) *non-alphanumeric printing character*

When the escape marker is followed by any non-alphanumeric printing character, it causes that character to be treated in its own right as part of the text, even if it is defined to have some significance for formatting. This can be used to prevent a punctuation symbol from being treated as a sentence terminator; to prevent a right bracket from being recognised as a scope terminator; to prevent a character defined as a single-character directive from being recognised as a directive; and finally to allow the printing escape marker itself to appear in the text.

(b) *space*

This convention is used to indicate that a source space is to cause a fixed width spacing effect in the printed document, even when composing mode is in force. It can be used to include additional spacing in a context where multiple spaces would otherwise be ignored, or to prevent recognition of a word boundary.

(c) *line-end*

This convention is used to indicate that a line-end in the source is to be treated as significant, even when line-filling is enabled.

(d) *other control character*

[Reserved for future extension]

Tabulate

`@T(`*tab-number*`)`

The T directive causes the horizontal position on the current line to be set to the value of a selected tab stop. The new position may be to the right or to the left of the existing position.

The Tabulate operation implies a word break and establishes a new left fix-point for subsequent text on the same line.

A special form of the T directive (T*) causes the selected position to be established as a continuing indentation point.

Absolute tab stop selection

In this case the argument to the T directive specifies an explicit tab stop. Tab stop 0 (zero) always identifies the left margin position. The first settable tab stop is number one. For example:

`@T1`	set horizontal position to tab stop 1
`@T1 (a) @T3 Note`	print `(a)` at tab stop 1, then `Note` at stop 3

Relative tab stop selection

In many cases, it is more convenient to specify a tab stop relative to the current line position. This is done by the T+ (tab forward) and T- (tab backward) variants of the T directive. For example:

`@T+1`	tab forward once
`@T-2`	tab backward twice

1. Tabbing forward means moving to the first tab stop set to a position to the right of the current position.
2. Tabbing backward means moving to the tab stop one before the first tab stop set at or to the right of the current position.
3. These operations are independent of how the current position was established.

Alignment

In addition to selecting a new horizontal position, the T directive may also be used to specify the alignment of the preceding text on the line. This is done by following the T with an alignment code. The text affected is all the text from the start of the line or the last horizontal positioning effect (the left fix-point) up to the T directive. The possible cases are:

`@T<`	Left align (to left fix-point)
`@T>`	Right align (to tab position)
`@T^`	Centre (between left fix-point and tab position)
`@T<>`	Justify (between left fix-point and tab position)

The alignment code immediately follows the letter T, preceding either of the relative symbols (plus or minus) and any argument.

Indentation

If the letter T is followed by an asterisk (*), the effect is to cause the selected position to be established as the current left indentation position. That is, as well as having immediate effect, the position will be automatically set as the initial line position when subsequent line breaks are made.

Relative forms (`T*+` or `T*-`) may be used.

For example:

`@T*1`	move to tab 1 and set this position as indentation position
`@T*-`	tab backward and set this position as indentation position

The T* directive may take a second argument, following a comma, to specify a second tab stop to be used to set a right indentation position. The right indentation position determines the effective line width for formatting purposes. The position must be within the current page dimensions. For example:

`@T*2,6`	format the following text between tab 1 and tab 6

The effect of the T* directive may be scoped. The effect continues until changed by another T* (or X*) directive, or pre-existing indentation positions are restored at the end of a containing scope, or block.

Tab setting

`@T`=*pos1,pos2,....*
`@T`(*tab-number*)=*pos1,pos2,....*

Tab positions are defined by a variant of the T directive which is distinguished by the occurrence of an equals-sign following the T (and optional tab number). The equals-sign is followed by a list of values for consecutive tab positions. Any tab stop may be set to any position. If no tab number is given, tab stop 1 is taken as the starting-point; otherwise the number given is taken as the starting tab stop. For example:

`@T=1",3.5",4"`	set tab 1 at 1", tab 2 at 3.5", tab 3 at 4"
`@T4=7"`	set tab 4 at 7"

As an alternative to specifying tab stop positions in inches, they can be given as column numbers (counting from zero at the left margin). In this case, the numbers given are interpreted as multiples of the column size interpreted in terms of the current font at the time of the definition. For example:

`@T=8,16,24,32,40`	set tab 1 at column 8, tab 2 at column 16, ...

Tab positions may also be specified relative to the preceding tab position by including a leading plus-sign ('+'). For example:

`@T=1",+0.2",+0.2"`	set tab 1 at 1", tab 2 at 1.2", tab 3 at 1.4"

Stop positions not included in a tab setting list retain their existing values

3. PROTOCOL SYNTAX AND SEMANTICS

3.1. Source data and Printed Document

In describing the LAYOUT protocol, it is necessary to make distinctions between the data stream which is presented to the printer and the printed document which that data stream causes to be printed. The two are representations of the same information in different forms. The first is entirely textual, utilising a limited character set; the second is essentially graphical, with a variety of character forms and pictorial effects. For the definition of the syntax of the protocol, it is the first of the two - the data stream sent to the printer - which is involved. It is important to make the distinction, since the data sent to the printer has its own line-structure and spacing effects, which may, in general, be quite different from those of the document created. Where the distinction between the two representations needs to be emphasised, the terminology *source data* and *printed document* will be employed.

3.2. Source data character set

The character set pre-supposed by the protocol is the American Standard Character-set for Information Interchange (ASCII), which may be either the basic 7-bit set or the extended 8-bit set. This does not imply any fixed assumptions about the graphical rendering of individual characters when they appear in the printed document. What is fixed is the interpretation of a certain range of characters as *control*, rather than *printing* characters, and the classification of certain characters as *letters*, *digits*, *punctuation symbols*, *brackets*, and the like. This interpretation is significant for the control information in the protocol.

When extended ASCII is used, the additional 128 character codes (128_{10} to 255_{10}) are regarded within the protocol as extensions of the printing character set; none of them is interpreted as a control character, letter, digit or bracket.

Two of the important characters in the source character set are the *space* (Ascii 32_{10}) and whatever control character (or character pair) is used to end a line. The latter is selectable as a configuration option. Within this document, it will be referred to as the *line-end* character. By contrast, the operation of terminating a line in the printed document will be referred to as *line break*.

Underline

@U(*width,position*)

The U directive is used to create underlining effects. All text and intervening spaces which appear between the point at which underlining is switched on and the point at which it is switched off are underlined by the placing of a continuous rule under the base-line.

The arguments determine the width and the position of the rule. The default values are determined according to the descender height of the font selected when underlining is switched on. The default width is 1/10 of the font descender height and the position below the base-line is 1/2 of the descender height, with the rule extending downwards from this position.

If either of these arguments is specified explicitly, it may be given as an absolute measure (in inches) or as a number, which is interpreted as a multiple of the descender height.

Specifying a zero width disables underlining.

The effect of the U directive may be delimited by following it by one of the left scope brackets [or {. In that case, the pre-existing state of underlining is restored at the end of the scope. If the U directive is not scoped, the effect persists until the state is changed by another U directive (or a previous state is restored by the ending of a containing scope, or block).

The U directive does not imply any kind of break.

Examples

@U	Default underlining
@U0.2,1	Underlining with bigger rule and lower
@U0.01",2	Underlining with 0.01" rule, lower again

Value

@V(*number*)

The V directive provides a means of inserting a numeric value into the text at the point at which the directive occurs. The argument is an arithmetic expression, usually a variable or compound expresion. The current value of the expression is treated as if it had appeared in the source data. By default the number is presented in free format decimal form.

The V directive does not imply any kind of break.

Examples

@V(pageno)	the value of variable *pageno*
@V300*6/8	the value 225

X-set

`@X`(*position*)

The X directive provides a means of explicitly setting the current horizontal position X to any desired position within the width of the current page. There is no change in the vertical position Y.

The X directive implies a word break, and establishes a new left fix-point (see *Alignment* below).

A special form of the directive (X*) can be used to establish a position for continuing indentation.

Absolute horizontal positioning

In this case the argument gives the required new value of X. It may be given in inches, for example:

`@X2.25"`	set X to two and a quarter inches from left of page
`@X1/3"`	set X to one third inch from left of page

Alternatively, the argument may be a simple value, in which case it is interpreted as a column number (counting from zero), that is as a multiple of the current column-size. For example:

`@X30`	set X to column position 30

Relative horizontal positioning

If the letter X is followed by the symbol plus (+), the argument indicates a movement forward from the current position. If the letter X is followed by the symbol minus (-), the argument indicates a movement backward from the current position. Interpretation of the argument is as for the absolute case.

Alignment

In addition to selecting a new horizontal position, the X directive may also be used to specify the alignment of the preceding text on the line. This is done by following the X with an alignment code. The text affected is all the text from the start of the line or the last horizontal positioning effect (the left fix-point) up to the X directive. The possible cases are:

`@X<`	Left align (to left fix-point)
`@X>`	Right align (to new position)
`@X^`	Centre (between left fix-point and new position)
`@X<>`	Justify (between left fix-point and new position)

The alignment code immediately follows the letter X, preceding either of the relative symbols (plus or minus) and the argument.

2.6. Current environment

The interpretation of the meaning and effect of control statements in the protocol depends on the *environment* current when the statements are executed. The different aspects of the current environment can be classified under three main headings.

Definition environment

The protocol contains a basic set of built-in capabilities which are always available for use. These may be extended by adding facilities through definition of extra fonts and macros. Some of these definitions may be fixed for a given installation, so that they are almost in the same category as the built-in capabilities. Others may be installed for a given batch of documents. Others may be specific to a single document, or a part of it.

To provide for the management of groups of definitions in a way that reflects the different levels at which various groups are required, the protocol provides a means of imposing a nested block structure on the data which allows definitions to be localised to a given block. Any definition made within a given block ceases to exist at the end of that block.

Special blocks, called document blocks, allow groups of definitions to remain in existence for a sequence of separate documents which require the same global definitions.

Current page

The current page is characterised by the position of the page within the containing page, the dimensions of the page, and its orientation. Collectively, these attributes allow positions within the page to be mapped to absolute positions within the physical page. These values are fixed when the page is opened and do not subsequently change.

There are also two dynamic state variables associated with a page which determine the current position within the page. These are X for the horizontal position and Y for the vertical position. Their values are automatically updated as part of the effect of many formatting operations.

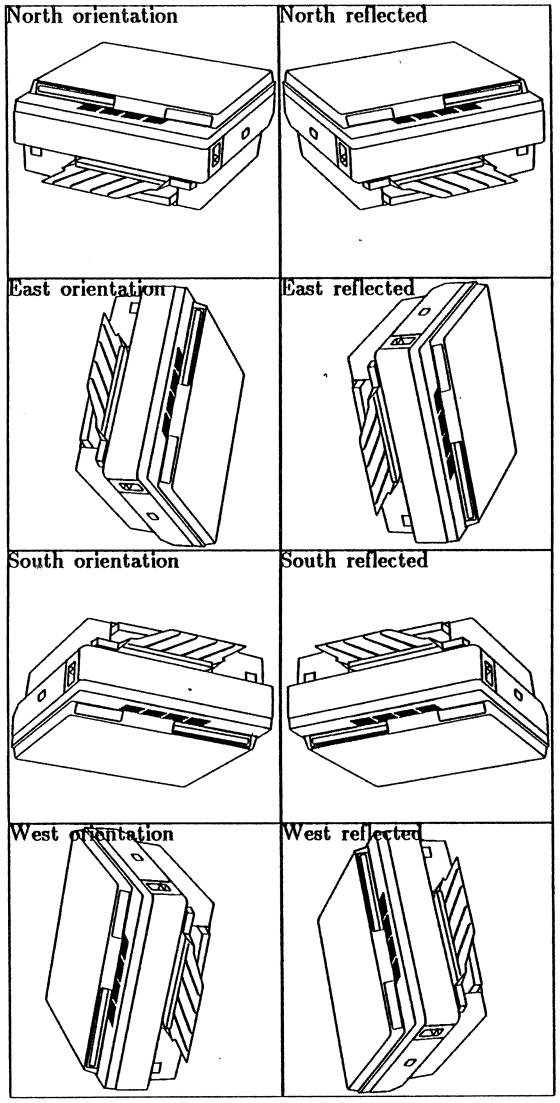
The page dimensions also determine the boundaries for printing and drawing operations pertaining to the page. The protocol defines that drawing operations are limited (clipped) precisely at page boundaries, whereas some relaxation is permitted for character placement (as described in Chapter 4: Fonts).

Formatting state

The current formatting state is characterised by a set of *formatting state variables.* These include mode variables which dynamically reflect formatting options which have been selected, such as current font, inclination and bolding factors, underlining, justification, and line-filling. There are other variables associated with the placement of characters on the line: C (Column-size), D (Displacement to base-line) and L (Line-increment).

For the general understanding and use of the protocol, it is not necessary to be concerned with the existence of most of these variables or the way they are utilised and updated. However, the precise definition of some formatting effects does require to be stated in terms of the values of these variables; and some of them can be accessed explicitly within the protocol.

At the end of a block, the formatting state variables revert to their values at the start of the block.



2.5. Relative interpretation

Within a page, all co-ordinate positions and directions are relative to the top left position of the current page. Similarly, the terms *left*, *right*, *top*, *bottom*, *horizontal*, *vertical*, *up*, *down*, *north*, *south*, *east* and *west* are to be understood in this relative sense throughout this document, unless stated otherwise.

Indentation

If the letter X is followed by an asterisk (*), the effect is to cause the selected position to be established as the current left indentation position. That is, as well as having immediate effect, the position will be automatically set as the initial line position when subsequent line breaks are made. For example:

`@X*1"` indent subsequent text by one inch

The X* form may take a second argument, following a comma, to specify a right indentation position. The right indentation position determines the effective line width for formatting purposes. The position must be within the current page dimensions. For example:

`@X*10,50` lay out subsequent text between columns 10 and 50

The effect of the X* directive may be scoped. The effect continues until changed by another X* (or T*) directive, or pre-existing indentation positions are restored at the end of a containing scope, or block.

Y-set

`@Y`(*position*)

The Y directive provides a means of explicitly setting the current vertical position Y to any desired position within the current page depth. There is no change in the current horizontal position X.

The Y directive implies line termination, but not a line break. That is, any partial line which is being composed is terminated and placed in the page image, but the line feed operation is not performed, so that the current position is not updated.

Absolute vertical positioning

In this case the argument gives the required new value of Y. It may be given in inches, for example:

`@Y2.25"` set Y to two and a quarter inches from top margin
`@Y1/3"` set Y to one third inch from top margin

Alternatively, the argument may be a simple value, in which case it is interpreted as a line number (counting from zero), that is, the value given is taken as a multiple of the line increment (see the B directive). For example:

`@Y30` set Y to line number 30

Relative vertical positioning

If the letter Y is followed by the symbol plus (+), the argument indicates a movement downward from the current position. If the letter Y is followed by the symbol minus (-), the argument indicates a movement upward from the current position. Interpretation of the argument is as for the absolute case.

System control

`@Z.xxxx`

The Z group of directives deal with a variety of control functions. This is an open-ended group, so that only some of the basic cases are described here. In all cases, this directive extends to the end of a source line, and it always implies a line break.

Among the functions covered by this group of directives is the selection of emulation modes which may be available as alternatives to the LAYOUT protocol. For example:

```
@Z.DAISY1(10,8)
@Z.DVI
```

The general effect is to switch into the selected mode so that subsequent source data is interpreted according to that protocol until a defined termination sequence is received. Interpretation in the new mode starts after the end of the source line containing the directive. The new mode inherits the current environment, in so far as it is relevant, and will in general modify aspects of the environment in the course of its activation.

The specification of what modes are available, the facilities provided by them, and the form of the termination sequence does not form part of the definition of the LAYOUT protocol.

Control of input sheets

`@Z.IN` *cassette-identifier*

The Z.IN directive selects one of a number of input cassettes or trays as the one from which subsequent sheets of paper are to be fed. Cassettes are identified by numbers starting from number 1.

Control of document output

`@Z.OUT` *copies, first-bin, last-bin*

The Z.OUT directive controls the output of pages from the printer. It can be used to select the number of copies to be printed and the bin to which the sheets of paper are to be directed. Once this selection has been made, it applies to all pages printed until another occurrence of the same directive, or the end of the current scope. The maximum number of copies which may be specified is determined by an installation selected limit. If a value outside this limit, or less than or equal to zero, or a fractional value, is given, it is treated as number 1.

The second and third arguments indicate the output bins to be used, counting from bin 1. The number of copies specified is directed to each bin in sequence.

Examples

`@Z.OUT 2,3,6`	Send 2 copies of each page to each of bins 3, 4, 5, and 6
`@Z.OUT 1,2,3`	Send single copies of each page to bins 2 and 3
`@Z.OUT 5`	Send 5 copies of each page to bin 1

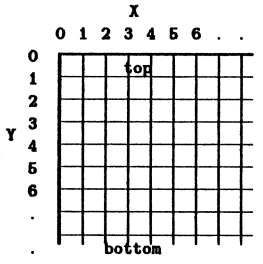
2.3. Virtual pages

The physical page size for the printer is one of a selection of possible paper sizes depending on printer type. Within one physical page, the protocol allows a number of sub-pages to be defined. When it is important to distinguish between complete pages and sub-pages, the terms *physical page* and *virtual page* will be used, but in general in the remainder of this document the term *page* will be used in the sense of *virtual page*.

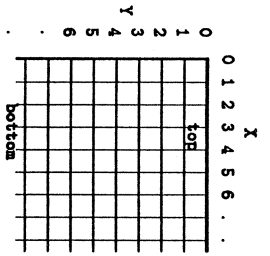
2.4. Orientation

When a new sub-page is defined, its position and dimensions are given in terms of the co-ordinate system of the current page - the page which contains the new one. In addition, a new *orientation* for the virtual page can be specified. The edges of a page must be parallel to the edges of the physical page, that is, the permitted rotations must be multiples of 90 degrees. The four basic orientations are illustrated below.

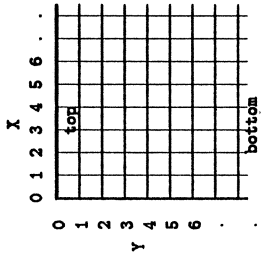
North (/\) orientation



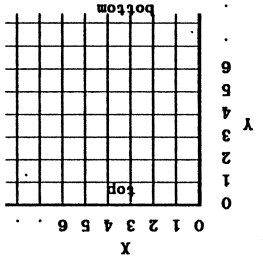
East (>) orientation



West (<) orientation



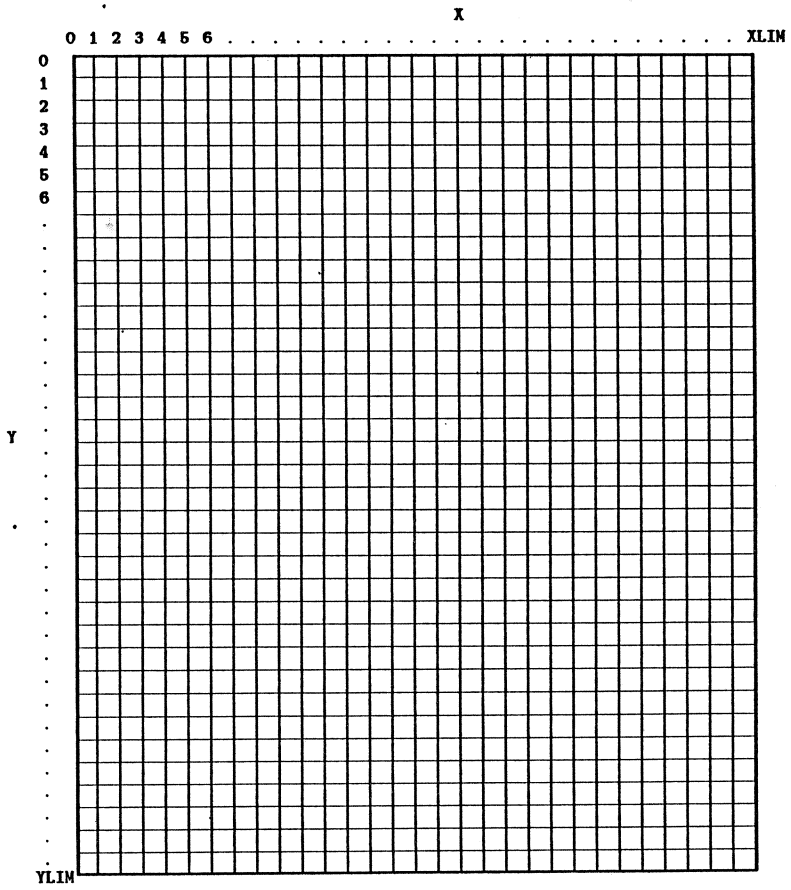
South (\/) orientation



It is also possible to specify that a page is to be *reflected*, giving a total of eight page aspects in all. The reflected page aspects are relevant mainly for graphics.

2.2. Co-ordinate system

In order to allow for the specification of positions on the page, down to the level of individual dots if required, it is necessary to have an underlying co-ordinate system expressed in terms of the printing device pixel resolution (typically 300 dots to the inch). It is rarely necessary in the protocol to specify positions directly in terms of pixels, since other units are available, such as columns and lines, or physical measures such as inches. The basic principles of direction and orientation apply equally to all the units. The co-ordinate system is based at zero and co-ordinate positions should be thought of as identifying the boundaries between pixels rather than the pixels themselves. The X co-ordinate values increase from left to right, and the Y co-ordinate values increase from top to bottom of the page.



Selection of pages

@Z.SELECT *page-range-list*

This command allows for selection of particular pages from the current document to be printed. Each element in the *page-range-list* specifies a single page number, or first and last page numbers separated by a colon.

For example, the command

`@Z.SELECT 1:4, 5, 9:11` would select pages 1 to 4 inclusive, 5, and 9 to 11 inclusive.

The numbers refer to physical pages, counting from the current page as number 1.

Displaying messages

@Z.MESSAGE *text*

The Z.MESSAGE directive causes a text message to be output on the printer display unit, if one is fitted. The message starts with the first non-space character after the identifier and continues to the end of the source line.

The maximum length of a message and the character set available for messages are installation dependent.

Suspending operation

@Z.PAUSE *text*

The Z.PAUSE directive causes suspension of processing at the printer to allow for operator interaction. Processing is resumed by manual request. A text message to be displayed in place of the standard 'Manual' message may be given.

Communications control

@Z.RSVP *character-code*

This command can be used to synchronise operational changes with the stage reached in processing the data stream. The character code specified by an arithmetic expression following the identifier is transmitted by the printer to the host at the time that the directive is executed.

Local file system operations

@Z.FILE

This group of commands provides for the manipulation of files held in the local filing system incorporated in the printer, if one is fitted. The details given apply to a basic single-drive diskette-based filing system. Each disk has a total capacity of 800 kilobytes and can accommodate up to 200 files (depending on size) organised into a maximum of 7 directories.

A file specifier consists of an optional disk label in square brackets followed by a directory name then a colon and a file name. A disk label is a single letter followed by a number in the range 1 to 255; a directory name is up to 6 letters or digits; and a file name is up to 12 letters, digits or periods. Examples of file specifiers are:

```
[B5]MAIN:FILE1
MANUAL:CHAP23.OLD
```

@Z.FILE *file-specifier*

The basic Z.FILE command by itself is used to include the content of the designated file in the source data stream. Inclusions may be nested, to a maximum of three.

@Z.FILE.WRITE *file-specifier*

This variant of Z.FILE is used to write data to the file specified The data starts on the following line and is terminated by a line containing just @Z. Any existing file of the same name in the specified directory on the disk involved is deleted once the new file has been successfully written.

@Z.FILE.DELETE *file-specifier*

This directive causes the designated file to be deleted.

@Z.FILE.INIT *disk-label;directory-names*

This directive is used to format, label, and initialise the directories on a disk. The disk must be already mounted in the disk drive. Any existing data on the disk is totally destroyed.

Initialisation is required before files can be written to the disk. Up to 7 directory names may be given, separated by commas.

Manual confirmation is requested before execution of this command.

2. FORMATTING ENVIRONMENT

The basic function of the protocol is to define images of pages to be reproduced on a high-resolution raster printing device. Ultimately, this implies defining for each of several million points on each printed page whether it is to be black or white. The protocol is implemented by a *controller* in the printer which receives and acts upon data in the form defined by the protocol sent by a host computer system.

Each printed page is created by the sequential execution of actions which are invoked by the character stream sent to the printer. Both the control sequences and the individual characters making up the text can be thought of as causing an action to be executed.

The controller first creates an internal image of each page of the document - the *page image* - and then causes the printing mechanism to reproduce this image on a sheet of paper. The basic operations involved in creating the internal image are analogous to committing ink to paper through familiar operations such as writing or typing. The principle is that of using a writing instrument: there is a current operating position which determines where the next information is to be written; the act of writing automatically alters the current position in a precisely defined way; and there are also explicit positioning actions which allow the current position to be set to any point in the image of a single page at any time. The page image is initially blank and the effect of writing is cumulative, which means that over-printing effects can readily be achieved. There is also a facility to erase areas of the page.

In practical terms, there are obvious differences between the internal page image and the printed sheet. Over-printing does not literally apply multiple layers of ink and erasure is total, for example. Conceptually, however, the distinction is unimportant.

2.1. Composition and Line-filling

Lines of text are *composed* in a fashion similar to type-setting. The printing characters which appear in the source data are formed up like a line of type and only when the composing unit is completed, is it placed in the page image. This allows decisions regarding spacing within the line and distance from the previous line, for example, to be taken on the basis of the total content of the line.

There is a major difference in the way source text is processed, according to whether *automatic line-filling* is enabled or not.

When automatic line-filling is *disabled*, plain text is treated in a fashion similar to the way it would be printed on a conventional computer printer. A default page format and character font are used. Consecutive *printing* characters and spaces are assembled in adjacent positions. Line breaks are caused by the appropriate *control* characters in the input stream. Over-long lines are truncated, not wrapped. Page breaks are caused either by reaching the bottom of the current page or by receipt of a Form Feed character. This mode is appropriate where the line structure of the final document is to follow that of the source. The mode can be enabled for a complete document, or for sections of it.

When line-filling is *enabled*, running text is automatically split up into lines, on the basis of how many words will fit within the available line space.

1.2. Printing effects

The basic printing effects which are available are:

character printing generation, placing and automatic spacing;

explicit positioning to an arbitrary position within the resolution of the printing device, using tabulation or direct addressing;

font selection including change of point-size and face within font family;

special effects such as underlining, bolding, and over-printing.

1.3. Formatting operations

The main formatting capabilities which are supported are the following:

page make-up controlling the position and orientation of the pages which make up the document, adding headings and other fixed background information;

virtual pages allowing the creation of multiple and nested rectangular formatting areas on a single sheet of paper;

text composition automatic computation of placement of characters, words and lines in the page image;

line-filling making lines up to a defined line length, or as near as can be achieved without splitting a word between two lines;

text alignment centring, justification, and right flushing;

page-filling making pages up to a defined page length;

indentation left and right, including nested indentation.

1.4. Definition and control features

Among the features available to the user to make it easier to utilise the full power of the protocol in a convenient way are the following definition and control facilities:

font definition to introduce new or derived character fonts;

variables pre-defined and user-declared, allowing values to be specified dynamically and stored for later use;

arithmetic expressions allowing the use of computed values as arguments;

macros allowing commonly required sequences to be defined under a mnemonic name;

format macros a special type of macro which allows a page format with associated headings and other fixed information to be defined and invoked on a continuing basis.

document structuring providing nested definition contexts and allowing formatting effects to be confined to a particular scope.

6. FORMATTING STATE VARIABLES

A Above-height
The value of this read only variable is the *above-height* of the currently selected font, or a font specified in parentheses immediately following the A. The value is a measure

B Below-height
The value of this read only variable is the *below-height* of the currently selected font, or a font specified in parentheses immediately following the B. The value is a measure

C Column-size
In an assignment to this variable, the value may be given as a measure or a number. The latter is interpreted as a multiple of the current value of C at the time of definition. If the assignment is terminated by a question-mark (eg `@C=0.2"?`), the value is *provisional*; otherwise it is *categorical*.

D Displacement
In an assignment to this variable, the value may be given as a measure or a number. The latter is interpreted as a multiple of the current value of D at the time of definition. If the assignment is terminated by a question-mark, the value is *provisional*; otherwise it is *categorical*.

E Escape-marker
The value assigned to this variable must be an integer (eg `@E=36` or `@E='$'`). If the value zero is specified, no additional character is recognised as an Escape Marker.

F Font selector
Within an arithmetic expression, the form F:*character-code*, or more generally the form *font-selector*:*character-code*, denotes the formatting width of the designated character. The value is a measure.

G Sentence-Gap
The value of this variable may be a measure or a number. The latter is interpreted as a multiple of the Column-size current at the point of occurrence of a sentence break.

H Page-Height
The value of this read-only variable is a measure.

I Indent-paragraph
The value of this variable may be a measure or a number. The latter is interpreted as a multiple of the Column-size current at the point of occurrence of a P directive.

L Line-increment
In an assignment to this variable, the value may be given as a measure or a number. The latter is interpreted as a multiple of the current value of L at the time of definition. If the assignment is terminated by a question-mark, the value is *provisional*; otherwise it is *categorical*.

P Point-size
The value of this read only variable is the *point-size* of the currently selected font, or a font specified in parentheses immediately following the P. The value is a number.

S Space-width
The value of this read only variable is the *space-width* of the currently selected font, or a font specified in parentheses immediately following the S. The value is a measure

T Tab()
In an assignment to tab-stop variables, each value may be given as a measure or a number. The latter is interpreted as a multiple of the current value of Column-size at the time of definition.

U Unit
This variable specifies the measure to be taken as the unit of horizontal movement for the Right-move directive. The default value is the width of the device pixel, typically 1/300".

W Page-Width
The value of this read-only variable is a measure.

X X-position
The value of this read-only variable is a measure.

Y Y-position
The value of this read-only variable is a measure.
During the process of composition, the values of the variables X and Y are provisional, since they are subject to later adjustment as a result of alignment or page-filling.

1. INTRODUCTION

The LAYOUT document formatting language is a protocol for defining page images for reproduction on a printing device. The protocol defines a set of control sequences which are embedded in the text to be printed, in order to create special printing effects and guide the way the text is formatted. In the design of the protocol, emphasis has been placed on compactness, in order to minimise the overhead which has to be added to plain text to achieve formatting effects.

LAYOUT is not a general-purpose programming language, and the computational and control facilities provided in it are geared to the special requirements of text formatting and image creation.

This document is a reference manual and necessarily includes a fair amount of technical detail. The examples given are chosen to illustrate aspects of the syntax and interpretation of the protocol, rather than to show how to achieve useful effects.

1.1. Levels

The protocol is designed for use at a number of levels. It can be used at a low level appropriate to output from word-processing or text-formatting software. In this case, most of the formatting and placement decisions are taken by the host software and the function of the protocol is simply to communicate these to the printer. The relevant aspects of the protocol for this level of use are the *printing effects.*

The protocol can also be used in such a way that most of the formatting and organisation of the text is delegated to the printer. In this mode of use, the protocol overhead is generally less than for low-level use and data for the printer can be produced directly by the user without the need for specialised software. The relevant aspects of the protocol for this level of use include the *formatting and control features.*

Appendix 1: Character coding

Basic coding

	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0				0	@	P	‘	p						—		
1			!	1	A	Q	a	q			¡	–	`		Æ	æ
2			"	2	B	R	b	r			¢	†	´			
3			#	3	C	S	c	s			£	‡	ˆ		ª	
4			$	4	D	T	d	t			/	·	˜			
5			%	5	E	U	e	u			¥		¯			ı
6			&	6	F	V	f	v			ƒ	¶	˘			
7			’	7	G	W	g	w			§	•	˙			
8			(	8	H	X	h	x			¤	¸	¨		Ł	ł
9			)	9	I	Y	i	y			‘	„			Ø	ø
10			*	:	J	Z	j	z			“	”	˚		Œ	œ
11			+	;	K	[	k	{			«	»	¸		º	ß
12			,	<	L	\	l	|			<	...				
13			-	=	M	]	m	}			>	‰	˝			
14			.	>	N	^	n	~			fi		˛			
15			/	?	O	_	o				fl	¿	ˇ			

Symbol coding

	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0				0	≅	Π	¯	π				∘	ℵ	∠	⋄	
1			!	1	A	Θ	α	θ			ϒ	±	ℑ	∇	⟨	⟩
2			∀	2	B	P	β	ρ			′	″	ℜ	®	®	∫
3			#	3	X	Σ	χ	σ			≤	≥	℘	©	©	⌠
4			∃	4	Δ	T	δ	τ			/	×	⊗	™	™	⌡
5			%	5	E	Y	ε	υ			∞	∝	⊕	Π	Σ	⎛
6			&	6	Φ	ς	ϕ	ϖ			ƒ	∂	⊘	√	⎛	⎝
7			∍	7	Γ	Ω	γ	ω			♣	•	∩	·	⎜	⎜
8			(	8	H	Ξ	η	ξ			◇	÷	∪	¬	⎝	⎛
9			)	9	I	Ψ	ι	ψ			♡	≠	⊃	∧	⎡	⎤
10			*	:	ϑ	Z	φ	ζ			♠	≡	⊇	∨	⎜	⎜
11			+	;	K	[	κ	{			↔	≈	⊄	⇔	⎣	⎦
12			,	<	Λ	∴	λ	|			←	…	⊂	⇐	⎛	⎝
13			-	=	M	]	μ	}			↑	|	⊆	⇑	⎨	⎬
14			.	>	N	⊥	ν	~			→	—	∈	⇒	⎝	⎛
15			/	?	O	_	ο				↓	↵	∉	⇓	⎜	

LAYOUT

FORMATTING LANGUAGE

REFERENCE MANUAL

1. INTRODUCTION
2. FORMATTING ENVIRONMENT
3. PROTOCOL SYNTAX AND SEMANTICS
4. FONTS
5. INDIVIDUAL DIRECTIVES
6. INDIVIDUAL STATE VARIABLES

APPENDICES

Version 1.1 December 1985

Katty Humphry

University of Edinburgh

Department of Computer Science

LAYOUT

FORMATTING LANGUAGE

REFERENCE MANUAL

Reference Manual CSR - 196 - 1985

James Clerk Maxwell Building,
The King's Buildings,
Mayfield Road,
Edinburgh,
EH9 3JZ.

Version 1.1 December 1985