UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 43

(February 1984)

Title:

A Proportional Spacing Version of LAYOUT: Notes for Inexperienced LAYOUT Users

Author: John M. Murison

Contact: Advisory Service

Software Support Category: C

Synopsis

This Note explains how to use a new version of the text formatting program LAYOUT. It does not assume any previous experience of LAYOUT.

The Note is based on the "Layout User's Guide (Revised specification)", August 1983, by Hamish Dewar, Department of Computer Science, University of Edinburgh. It has been prepared using the version of LAYOUT it describes, in a proportionally spaced Gothic font. In addition, the text has been right-justified.

Keywords

Document formatting, text formatting, LAYOUT

Edinburgh Regional Computing Centre
James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

The need for formatting programs

For run-of-the-mill purposes, there is no need to run a computer generated document through a text formatter before printing it. This applies to documents which have no special layout requirements or where the desired layout can be created by use of a standard context editor. However, in the case of documents to be printed on printers with multiple font and proportional spacing facilities, this approach is inadequate, since it does not permit the full capability of the printer to be utilised. In addition, it is not the most convenient approach for documents that are likely to go through several drafts or require regular updating. Assistance in meeting both of these requirements is provided by a type of computer program known as a paginator or formatting program. This document describes one such program, called LAYOUT.

Special printing effects

Context editors can work only with the standard (limited) character set available on a computer terminal and ordinary line printers. This standard computer character set (known as ASCII) cannot handle underlining, or subscripts and superscripts, or delicately variable spacing, or proportionally-spaced and multiple fonts.

Text formatters such as LAYOUT provide access to such facilities by defining formatting commands which may be incorporated in the document at appropriate places. These commands are not printed in the final document but are interpreted by the formatter to produce the required effect. In LAYOUT, some formatting commands are represented by single characters, called markers, and some by multi-character directives. The characters to be used as markers can be selected within the text file; it is customary to choose little-used punctuation symbols.

For example, there is a marker which is used to achieve the effect of underlining. A common choice for this purpose is the "_" character. Text to be underlined is enclosed between two occurrences of this symbol; that is, the effect of the symbol is alternately to switch underlining on and off. For example, including "He _will persist_ in pronouncing it '_Dun_das'" in the source file would cause "He will persist in pronouncing it 'Dundas'" to appear in the final document.

As noted, the choice of character to be used for markers is at the discretion of the user, and the choice can be switched at any point in the text file. There is also an 'escape' mechanism, using the escape marker (by default "$"), which causes a marker to be treated as an ordinary text character. Thus including "$_" in the text causes "_" to appear in the final document. Similarly, to cause the escape marker itself to appear in the text, it is necessary to duplicate it, including "$$" to obtain "$".

Main formatting operations

The other aspect of document production which is handled by text formatters is that of laying the text out on the page. The main functions which can be automatically carried out are the following:

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

justification, that is inserting extra spacing to produce an even right margin;
page-filling, that is making pages up to a standard length, and including headings and footings;
applying selected conventions for the layout of paragraphs and subsections;
indentation of sections of the text;
formatting of tabular information, so that the columns line up.

Most of these operations are optional; if they are not wanted, there are ways of switching them off by means of directives included in the text. Directives take the form of an escape marker (by default, the dollar-mark), followed by a mnemonic for the operation, optionally followed by a number. The mnemonic for all of the built-in operations is a single letter. It is immaterial whether the letters are typed in upper-case or lower-case, though upper-case is usually employed in this document. "$" is used as the escape marker in examples in this document. Examples of complete directives are "$B4" and "$L0".

The directive "$A" (Assign) has a different form. The Assign directive is used to alter the values of the basic parameters which control the formatting process. For example, there is a parameter ("LINE") which defines line width and a parameter ("JUST") which controls whether justification is performed. There are also parameters corresponding to each of the marker functions which define the characters to be used for each case. These parameters reflect the particular choices of user-selectable options which are required for a particular document or part of a document.

Line-filling

When line-filling is in force, the formatter ignores the line structure of the original text file, hereafter called the <u>source</u> file. A line-break separating two words is treated as exactly equivalent to a space between two words. Lines for the final document are assembled by taking words in sequence until the next word cannot be fitted onto the current line. At that point, a line-break is made and justification is applied if selected.

Two advantages stem from this mode of operation. First, the source file can be prepared -- and, more importantly, changed -- without too much concern for preserving uniform line lengths. Secondly, a decision to use a different line-length in the final document can be implemented simply by altering one statement in the source file. All the consequential changes in the final document, which may be substantial, come about automatically.

Words and sentences

So far as line-filling is concerned, a word is a sequence of characters delimited fore and aft by any of the following: a space, a line-break, or a directive. If more than a single space is included at any point in the source text, the additional space(s) are <u>ignored</u>, except when at the start of a line. A sentence is a sequence of words starting with a capital letter and ending with any of the following punctuation symbols: ".", "?", or "!". The only relevance of the sentence as a unit is that LAYOUT inserts extra space at sentence boundaries, the amount being user-selectable.

Justification

If the justification option is selected, LAYOUT inserts additional spacing as necessary into a line to bring its length up to the prevailing maximum line length. This creates a flush right margin in running text.

Justification is applied only to lines which have been automatically filled.

Avoiding line-filling

Obviously it may be inappropriate to have line-filling applied to the whole of a document; there are points at which line breaks are wanted in any case (irrespective of whether the next word could be fitted on the previous line). As line-filling is applied only to what can be regarded as running text, in many cases it is unnecessary to take any special action to avoid undesired line-filling. In particular, LAYOUT does not attempt to apply line-filling across a blank line, nor will it move words to the previous line from a line starting with a space. Furthermore, space characters at the start of a line are retained in the final document.

There are cases where it may be most convenient to create the desired format for a number of lines explicitly in the source file and tell LAYOUT to leave it alone. This done by means of the "$L" (Lines) directive.

Underlining

In order to cause part of the text to be underlined, it should be enclosed between occurrences of the underline marker (typically "_"). Thus the effect of the marker is alternately to switch underlining on and off. There is also a word underline marker (typically "%") which switches on underlining on a word basis, that is, until a non-letter is reached.

The $L directive

The "L" stands for 'lines' (or 'leave it alone') and this directive is used to indicate that a certain number of lines immediately following the line containing the directive are to be taken as they stand. The number involved is specified after the letter "L". For example "$L5" means 'leave the next 5 lines alone' and "$L1" means 'leave the next line alone'. As a special convention for this directive, a value of "0" (zero) may be specified, and this means 'leave all following lines alone until another directive causing a line-break is reached'. The zero convention is useful when the number of lines is quite large, since it saves counting them.

There are several additional options which may be selected in conjunction with "$L", by appending mnemonic letters to the directive. The letter "U" requests that the complete line(s) should be underlined, the letter "B" that they should be printed in bold-face, and the letter "M" (for middle) that they should be centred. For example, the following source lines:

```
$L2UM
First line of pair
Second (and last) line of pair
```

would produce the following output:

<u>First line of pair</u>
<u>Second (and last) line of pair</u>

The $B directive

While it is possible to cause blank lines to appear in the final document simply by including blank lines in the source text, it can be more convenient in some cases, particularly if the number is large, to make use of the "$B" directive. The "B" stands for 'blank lines' and this directive is used simply to cause a specified number of blank lines to be included at the point at which the directive occurs. Again, the number is typed immediately after the directive letter. Thus "$B4" means leave 4 blank lines. Note that "0" (zero) has its natural significance for this directive, and that "$B0" causes a line break but does not introduce any blank lines.

The $P directive

At a point in the text where it is required to start a new paragraph, the user may find it most convenient simply to leave a number of blank lines (typically two, one or none) and to indent the first line of the paragraph by including a few spaces. Again, however, for some purposes, it may be preferable to use a specific LAYOUT directive to mark the start of a paragraph. This is the "$P" directive. The effect is:

(a) a line break is made
(b) a page break is made if fewer than two lines are available on the current page
(c) a specified number of blank lines are left (unless a page break is made)
(d) the first line of the paragraph is indented by 3 spaces

The number of blank lines to be inserted is indicated by a number immediately after the "P"; "0" (zero) is permissible. Also, there is an assignment statement which may be used to alter the number of spaces to be inserted at the start of the paragraph, if some number other than three is preferred.

The $N directive

As with line-filling, the normal action of LAYOUT is to put as many lines on a page as will fit within the defined page size (which is user-selectable). The "$N" (Newpage) directive may be included in the source at any point where a page break is wanted, whether or not further lines would fit on the current page.

The $V directive

If pagination is performed in a simple-minded fashion, it can easily lead to unfortunate page breaks, with, for example, a heading separated from the start of the text to which it applies. For this reason, it can be prudent to include in the source file at appropriate points, "$V" (Verify) directives which cause a page break to be made if fewer than a certain number of lines remain unused on the current page. The "V" is followed by the number of lines to be checked for. Often, Verify directives are combined with other directives in a stereotyped group to give effect to the user's preferred way of, for example, starting a new section of the text. The following source line might be used to introduce a section heading, to be underlined and preceded by two blank lines: $B2 $V4 $L1U

Note that in this context the Verify directive should follow the Blank directive, whereas a Verify followed by a Blank directive may be used to guarantee that a certain number of blank lines should be left and these should not be broken by a page boundary.

Tabular data

Sometimes in designing the layout of a document it is a requirement to cause text to be aligned to a particular position on a line, perhaps to give a standard appearance to header lines or when preparing a table in which columns have to be lined up under each other. One way of doing this is to put in a "$L" directive to prevent any unwanted line-filling and lay the text out with additional spacing in the source file, 'manually' as it were. With a screen editor, this approach may be at least as convenient as using the special LAYOUT facilities provided for the purpose. Initially, most people prefer to adopt the 'manual' approach.

However, there are two considerations which may make it desirable (in due course) to move on to the LAYOUT facilities. The first is that, where several lines are laid out according to the same format, it is possible to adjust the format by altering a single assignment statement rather than having to modify each case individually. The second is that, when proportional-spacing fonts are to be used in the final document, columns lined up by the addition of spaces in the source file will not necessarily be lined up in the final document.

The way in which fixed positioning effects are achieved in LAYOUT depends on the familiar concept of 'tab' settings. There are two aspects to this: setting the tab positions, and using them. Setting tabs is achieved by including in the source file an assignment like the following:

```
$A TAB = 10,20,40
```

which would have the effect of establishing tabs at column positions 10, 20, and 40. Any number of positions may be specified, in ascending order. Subsequently, a particular tab position may be selected by including a "$T" directive at the appropriate point in the text. The "T" is followed by the tab number, counting from 1. Note, however, that columns are numbered from zero. Thus, with the assignment cited above, the following source lines:

```
$L0
$T1 1. $T2 New equipment $T3 11.30
$T1 2. $T2 Maintenance $T3 12.15
$T1 3. $T2 LUNCH $T3 12.45
```

would produce the following output:

```
          1.        New equipment       11.30
          2.        Maintenance         12.15
          3.        LUNCH               12.45
```

As $T does not cause a line break in the output, the $T directives in this example do not terminate the effect of the $LO directive.

The form $T+1, $T+2, etc. is permitted: it causes the text following to be placed at the next tab position, or next-but-one tab position, etc.

As the most common requirement for tabbing is simply to move to the next setting along the line, a special 'one tab' character, normally '|', can be defined. Thus another way of preparing the above source lines would be:

```
$A ONETAB='|'
  :
  :
$LO
| 1. | New equipment | 11.30
| 2. | Maintenance | 12.15
| 3. | LUNCH | 12.45
```

The 'one tab' symbol is read by LAYOUT as '$T+'. Thus, for example, '|2' would be interpreted as '$T+2'. If no digit follows the 'one tab' symbol then 1 is assumed (as in the example above).

Indentation

The method used to cause a section of text to be indented also makes use of tab settings. That is, the amount of indentation required is expressed as 'to a specified tab setting'. This is done by including in the source file at the start of the section of text in question an assignment to INDENT of the number of the tab required. At the point where indentation is to stop, an assignment of '0' (zero) to INDENT has to be included. All the lines in between will be indented.
For example, the following source lines:

```
$A TAB=4; INDENT=1
All the material typed in following the assignment of a non-zero value
to INDENT is indented to the corresponding tab setting, and this
continues until a different assignment to INDENT is made.
Here the first tab position is defined to be at column number 4
(the fifth column);  INDENT is then defined to refer to this first
tab position.
Note that it is not necessary to have an assignment to TAB
immediately before an assignment to INDENT, as is done in the example;
once appropriate
tab settings have been defined they are available for use until changed.
$A INDENT=0
```

would cause the following lines to appear in the final document:

> All the material typed in following the assignment of a non-zero value to INDENT is indented to the corresponding tab setting, and this continues until a different assignment to INDENT is made. Here the first tab position is defined to be at column number 4 (the fifth column); INDENT is then defined to refer to this first tab position. Note that it is not necessary to have an assignment to TAB immediately before an assignment to INDENT, as is done in the example; once appropriate tab settings have been defined they are available for use until changed.

Indentation is often used in order to present a sequence of enumerated points. It may be a requirement that the numerals or letters or whatever used to label the sections of text should not be indented, or should be indented less than the body of the text. For example:

1. This is main point number one. Like other main points it is indented to tab setting 2, but the identifying number '1.' is 'unindented' to tab setting 1.

2. This is main point number two. It starts off just like the previous one, but now, to make life interesting, we decide to include a couple of sub-points or subsections here.
   (a) Sub-point one. The text part is indented to tab setting 3, but again the label '(a)' is unindented.
   (b) Sub-point two. Same again.

3. This is main point number three, so we have returned to the same pattern as for main points one and two.

To achieve this effect, the source file could be prepared as follows:

```
$A TAB=5,11,19
$A INDENT=2
$T1 1. | This is main point number one. Like other main points
it is indented to tab setting 2, but the identifying number '1.'
is 'unindented' to tab setting 1.
$B $T1 2. | This is main point number two. It starts off just like
the previous one, but now, to make life interesting, we decide to
include a couple of sub-points or subsections here.
$A INDENT=3
$B0 $T2 (a) | Sub-point one. The text part is indented to tab setting
3, but again the label '(a)' is unindented.
$B0 $T2 (b) | Sub-point two. Same again.
$A INDENT=2
$B $T1 3. | This is main point number three, so we have returned to
the same pattern as for main points one and two.
$A INDENT=0
```

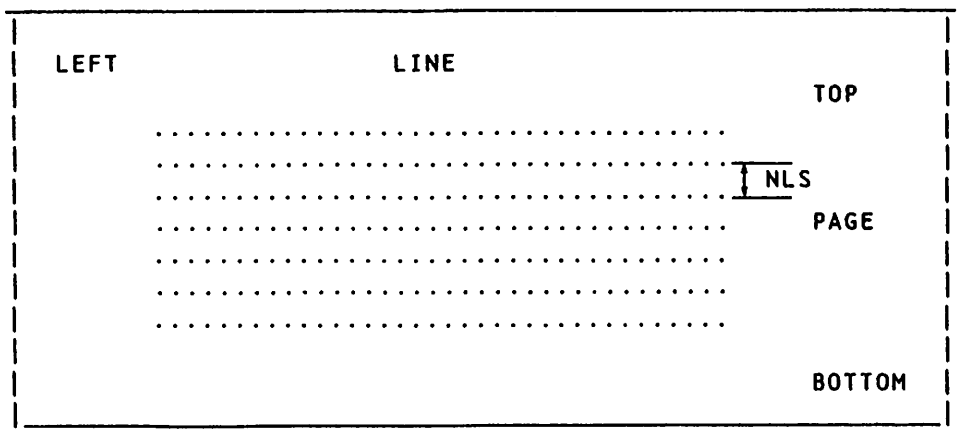
Apart from the use of multiple indentation levels, the main point to note is the use of the explicit tab selections to position the labels.

LAYOUT parameters

A number of examples have been given in the preceding sections of parameters to the formatting process whose values may be defined by assignment directives. This section provides further information about these and other parameters. There are default values to which the parameters are initially set, which are intended to be convenient for the most common requirements.

Some of the parameters define the physical format of the final document. Typically any required assignment of these would be made at the start of the source file and not subsequently altered. Other parameters, like the level of indentation, are more volatile, and are likely to be re-defined at regular intervals.

The parameters which define the physical format of the output document are indicated in the following diagram:



LEFT defines the left margin, while LINE defines the number of actual printing positions. Note that the intention is that LEFT should be used solely to control the placing of the text on the physical page; it should not be used for indentation. TOP and BOTTOM define the number of lines to be left at the top and bottom of the physical page respectively, and again PAGE defines the number of actual printing lines. NLS defines the vertical line spacing - i.e. the number of physical lines to be taken to print each line of text. When NLS is 1 (the default), TOP plus PAGE plus BOTTOM should add up to the intended physical page size (66 lines for standard line-printers).

The parameters described in the preceding paragraph can be assigned <u>fractional</u> values. The most useful application of this is to set

```
$A NLS=1.5
```

to achieve 1.5 line spacing rather than single spacing. The next paragraph is spaced in this way to illustrate the effect.

There is a parameter PAGENO (page number) which controls whether (and how) pages are numbered. Its default value is zero, which means no page numbering. If it is assigned a non-zero value, pages are numbered starting with the given value, and as successive page breaks are made, the number is increased by one. There is an additional parameter SECTNO (section number) which can be used to number sections (here understood to be a major unit of the document) and pages within sections. When page-numbering is requested, the number appears midway along the middle line of the bottom margin.

The parameter JUST controls whether lines are to be justified or not. Assigning the value 1 to it enables justification, while the value zero (the default) disables justification.

As mentioned earlier, there are also parameters corresponding to each of the special characters used in LAYOUT. With the exception of ESCAPE ('$'), none of these parameters is assigned a character by default. Assignment values take the form of a character enclosed within single quotes, rather than a numeric value, except that a value of zero may be specified to disable the facility altogether. For example, the assignment:

```
$A SUP='^'; SUB=0
```

would mean that the character '^' is to be used for marking superscripts and that no subscripting is required.

Multiple and proportional-spacing fonts

Where the output device provides more than one character font, it is necessary to have a means of indicating in the source text which font is to be used. This is done by including, wherever a change of font is required, a directive consisting of the escape marker followed by a font number. For example, the sequence "$3" selects font number 3. The exact significance of the font number will depend on the particular output device, and perhaps also which fonts are currently installed. Conventionally font zero is the standard or default font for the device, understood to be selected at the outset.

The availability of fonts of different sizes or proportional-spacing fonts, in which not all the characters are of the same size, creates a difficulty relating to the way in which line lengths and page sizes are specified. Measures along the line -- horizontal measures (like LINE and TAB) -- are presented in terms of character or column positions, and vertical measures (like TOP and PAGE) in terms of lines. How are these measures interpreted when not all characters are the same width and not all fonts are the same height?

This problem is compounded by the fact that different kinds of printer have differing capabilities in terms of the resolution of carriage or print-head movement. The typical daisy-wheel printer has a horizontal resolution of 1/120th of an inch and a vertical resolution of 1/48th of an inch, while many photo-typesetters operate with a resolution of 1/1000th of an inch.

The approach taken in LAYOUT is to permit the user to continue to express measurements in terms of columns and lines, but to translate these to device-specific units of horizontal and vertical movement, the translation being determined by the currently prevailing font selection. Specifically, the horizontal measures LEFT, LINE, SGAP, PGAP and TAB are scaled by the <u>width of the space character</u> in the currently selected font, as are values specified in $C directives. Values assigned to the vertical measures TOP, PAGE, BOTTOM and NLS are scaled by the overall font height. For simple ASCII devices, the scaling factors are unity.

It should be understood that the horizontal and vertical measures are <u>not</u> changed automatically by a change of font; it is only when one of the measures is explicitly assigned a value (by means of $A ...) that the currently selected font is relevant.

Author

LAYOUT was designed and implemented by Hamish Dewar, Department of Computer Science, University of Edinburgh.

Summary of main LAYOUT directives (by examples)

$L5	leave the next 5 source lines alone
$L1U	leave the next source line alone, but underline it
$L1M	leave the next source line alone, but centre it
$L1UM (OR $L1MU)	leave the next line alone, but underline and centre it
$L0	leave all following source lines alone until the next directive (or the end of the source file)
$B3	insert 3 blank lines
$B0	start a new line
$P2	start a new paragraph, inserting 2 blank lines
$T6	insert spacing up to tab setting 6
$C40	insert spacing up to column position 40
$N	start a new page
$V6	start a new page if within 6 lines of end of page

Summary of main LAYOUT parameters (with default values)

The 'DPLAY' defaults refer to a version of LAYOUT available on EMAS (see 'Access to LAYOUT on EMAS', below).

	standard defaults			DPLAY defaults
TOP	(2)	margin at top of page	) scaled	(0)
PAGE	(60)	usable page length	) by	(60)
BOTTOM	(4)	margin at bottom of page	) font height	(4)
NLS	(1)	line spacing	)	(1)
LEFT	(0)	left margin	)	(5)
LINE	(72)	line length	) scaled	(72)
SGAP	(2)	spaces between sentences	) by	(2)
PGAP	(3)	spaces at start of paragraph	) space width	(3)
TAB	(8,16,24,...,128)	current tab settings	)	(6,12,18,...,90)
PAGENO	(0)	current page number		(0)
SECTNO	(0)	current section number		(0)
JUST	(0)	justification selector		(1)
MARK	(0)	page separator selector		(2)
INDENT	(0)	indentation tab setting		(0)
ESCAPE	('$')	escape character		('$')
UNDER	(0)	underline start or stop character		(0)
WUNDER	(0)	word underline character		(0)
SUP	(0)	superscript marker (next character only)		(0)
SUB	(0)	subscript marker (next character only)		(0)
BOLD	(0)	bold-face start or stop character		(0)
ONETAB	(0)	tab character, equivalent to '$T+'		(0)

Check-list of special characters

Following the assignments

```
    $A WUNDER='%'; UNDER='_'; SUP='\'; SUB='/'; BOLD='~'; ONETAB='|'
```

the characters listed below will have the effects described. Note that '$' (Escape) is predefined; all the others must be assigned, as above, before they can be used.

```
%    underline following word (or remainder of word);  effect
               terminated by non-alphabetic not preceded
               by escape marker
_    start/stop underlining text
\    superscript marker (affects next character only)
/    subscript marker (affects next character only)
~    start/stop bold-face printing
|    insert spacing to next tab setting
$    escape -- when followed by a letter, introduces a directive
            -- otherwise prevents following character from having
               any special meaning
               (useful special case: $. -- period not ending sentence)
```

Changes from previous version of LAYOUT

Modifications:-

1. Columns numbered from zero, not 1.
2. No characters have special significance initially, except '$'.
3. Blank lines treated as significant.
4. Line starting with space treated as implying line break.
   Spaces at start of line treated as significant.
5. Underline treated as toggles, not single character markers.
6. Word underline facility changed.
7. No 'updated source' facility.

Additions:-

1. Support for proportionally spaced fonts and font selection.
2. Bold-face facility.
3. Single ONETAB character, equivalent to '$T+'.

Access to LAYOUT on EMAS

These notes are mainly intended for users of Philips GP300 printers, accessible from EMAS at ERCC by use of device names of the form '.DPnn', e.g. .DP15.

```
Command:OPTION(SEARCHDIR=CONLIB.GENERAL)            (once only)

Command:DPLAY(input1, input2 / draft, document)    (for .DP printer)
```

There can be two input files: 'input2' (optional) is read by LAYOUT first, then 'input1'. Thus, one can put standard parameter settings into 'input2', and document text into 'input1'. 'input1' is mandatory.

There are two output files (or devices).
'draft' is intended to be listed on a line printer or at interactive terminal. It gives an indication of how LAYOUT has formatted the text, but since proportional spacing character fonts, subscripts, etc. might have been used, the result on a line printer may look strange (since such effects cannot be accurately represented); however the draft output does enable one to see where line and page breaks have occurred. The default value for draft is .OUT. If it is not required, the form of the call must be

```
Command:DPLAY(..../.NULL,document)
```

'document' is intended to be listed to a GP300 printer.

This version of LAYOUT has several entry points; DPLAY is the one to use to generate text for a .DP printer. The other entry points are:

NLAYOUT	For fixed pitch ASCII printers - underlining achieved by use of Carriage Return and overprinting.
TTLAY	Same as LAYOUT, except that underlining is represented by setting the 8th bit of each underlined character.
SANLAY	For Sanders Printer - provides bold printing, underlining, superscripts, subscripts, variety of fonts.
DIALAY	For Computer Science Department Diablo daisy-wheel printer - provides bold printing, underlining, superscripts, subscripts.

GP300 Fonts

The fonts currently defined for the GP300 printer are detailed in User Note 50 and illustrated below.

0	Gothic 12cpi	The default font.
1	Gothic 10cpi	Slightly more spread out than font 0.
2	Gothic ps	The text of this note is mostly in this font, which actually gets more on a line than font 0.
3	**Gothic Bold 12cpi**	**There are two ways of getting Gothic bold text: either use fonts 0- 2 with the BOLD symbol, or use fonts 3- 5.**
4	**Gothic Bold 10cpi**	**Slightly more spread out than font 3.**
5	**Gothic Bold ps**	**Proportionally spaced text is easier to read!**
6	Courier 12cpi	I think this font looks rather cramped.
7	Courier 10cpi	That's much better.
8	Courier ps	Better still. Note that Courier ps takes up more space than Gothic ps (font 2).
9	Micro 12cpi	Rather small - but the symbols are still well-formed.
10	Micro 10cpi	Too spread out - looks a bit strange.
11	Micro 15cpi	Much better! Note that there is no Micro ps font available. Micro fonts are useful for superscripts and subscripts.

12	***ORATOR 12CPI***	***GOOD FOR HEADINGS. 12CPI IS RATHER CRAMPED. ORATOR FONT HAS CAPITALS ONLY.***
13	***ORATOR 10CPI***	***THAT'S BETTER! BUT STILL, A LITTLE GOES A LONG WAY.***
14	***ORATOR PS***	***THAT'S BETTER! BUT STILL, A LITTLE GOES A LONG WAY.***
15	Data 12cpi	This is a draft font - the printer goes at twice its normal speed, but the resolution is poorer.
16	Data 10cpi	Rather spread out, but it will give you the right line structure if you eventually want to use another 10cpi font.
17	Data 15cpi	Quite an attractive font, but could cause eye-strain if used too much.
18	Scientific 12cpi	|⊓⊔H┬┴─◊%.∽■±≡≈∽ ϱψεσυ™ ΡΨΕΣΥ∫ ατδζη ΑΤΔΖΗ
19	Scientific 10cpi	|⊓⊔H┬┴─◊%.∽■±≡≈∽ ϱψεσυ™ ΡΨΕΣΥ∫ ατδζη ΑΤΔΖΗ
20	Scientific ps	|⊓⊔H┬┴─◊%.∽■±≡≈∽ ϱψεσυ™ ΡΨΕΣΥ∫ ατδζη ΑΤΔΖΗ
21	Greek 12cpi	αβχδεφγιηκλμνοπ ΑΒΧΔΕΦΓΙΗΚΛΜΝΟΠ 1234 *+:;
22	Greek 10cpi	αβχδεφγιηκλμνοπ ΑΒΧΔΕΦΓΙΗΚΛΜΝΟΠ 1234 *+:;
23	Greek ps	αβχδεφγιηκλμνοπ ΑΒΧΔΕΦΓΙΗΚΛΜΝΟΠ 1234 *+:;

Justification was switched off when producing the above table. The rest of the document illustrates the use of a proportional spacing font with right justification switched on.

Note that other fonts may become available shortly, and that the 'data' fonts (15-17) may be withdrawn.