Layout

Layout is a text formatting language defining commands (called *directives*) which are embedded in the source text (the *manuscript*), and which control the way that the output (the *formatted text*) is produced.

This note describes *Layout* version 1.5, which is an intermediate version but the best for which the department has the source code. The only available document on this version of *Layout* ('Layout 1.5 Provisional User's Guide' by Hamish Dewar, Edinburgh University Computer Science Department, February 1984) is incomplete and slightly inaccurate, as well as not being widely publicised. It is hoped that this note will serve as a manual for *Layout* 1.5 and so make it easier to use.

A sample *Layout* 1.5 manuscript file (the manuscript for this document) is in the file Text:Lay.lay on the APMs.

1. Escape Marker

A basic characteristic of *Layout* is that the manuscript, in general, represents the text to be printed unless explicitly flagged otherwise. The method of flagging text as other than text to be printed (i.e. as a directive) is to precede it with the *escape marker*. By default, this is '$', the dollar sign, but if this is an awkward choice it can be redefined in the document.

2. Atoms, Words and Sentences

The *atom* is the basic unit of text recognised by *Layout*. Atoms are not split when line filling, and are not stretched when justifying the line. An atom is defined as a sequence of characters, delimited on both sides by one of the following:

- a space
- a newline
- an atom breaking directive

A word is similar to an atom, but it consists entirely of letters; that is, it is terminated also by any character other than a letter. It is relevant only to the *explicit words* directive.

A sentence consists of a sequence of atoms starting with a capital letter and finishing with a full stop, a question mark or an exclamation mark. The only relevance of sentences in *Layout* is that extra space is inserted between them, the actual amount of space being user-definable.

3. Spaces and Newlines

Within the scope of an *explicit lines* directive, both spaces and newlines within the text are treated as significant, and are preserved in the formatted text. Outside any such scope, line filling is enabled and, in general, spaces and newlines serve only to separate words and are treated equivalently; multiple spaces, for example, are treated as a single space. Any additional spacing effects are produced with directives. However, some commonly occurring requirements are covered by letting spaces and newlines be significant in certain situations:

- Blank lines (formed by two consecutive newlines) are preserved in the text; thus blank space can be produced simply by leaving blank lines in the manuscript.
- Spaces at the start of a line of the manuscript are significant, and are preserved in the formatted text. They also make the preceding newline significant, that is, they force a line break. Thus, a new paragraph can be started by leaving the required number of blank lines, then starting the first line of the new paragraph with several spaces.

Spaces are also treated as significant after explicit positioning directives, such as moving to a tab stop.

4. Directives

Directives can take several forms, as follows:

- Escape marker, followed by a single letter. This invokes a basic (built in) directive. The directive may be followed by one or more arguments, depending on the particular directive. The case of the letter is not significant.
- Escape marker, followed by a number. This selects a previously defined font. There can be up to 64 fonts, numbered from 0 to 63, of which 0 and 1 are predefined as horizontal and vertical fixed pitch (and rather grotty) fonts.
- Escape marker, followed by two or more letters. This invokes a previously defined macro. The case of the letters is again not significant. The invocation is implemented as straight text substitution; thus the macro itself needn't be syntactically valid as long as it becomes valid in context, and nested macro calls are allowed (up to a maximum of 16 levels).
- A single non-alphanumeric symbol. This invokes a previously defined

single-character macro (the term applies to the name of the macro, rather than its body); the invocation is implemented as above.

Note the possible ambiguity between a built in directive followed by text, and a macro call. This means that some way of terminating directive names is required. In the first three cases above, a single space after the directive will be ignored.

5. Arguments

Some directives take a numeric argument, usually specifying a measurement but sometimes specifying other items such as fonts. There is a default unit (such as lines, pixels, tab settings and so on) for the argument, dependent on the directive; the argument is specified as a multiple of this unit. Where non-integer values are relevant, the multiple can be expressed as a decimal, or as a fraction with the numerator and denominator separated by an oblique, '/'; the units may be changed to inches by following the argument with '"'.

Some directives allow the argument to be specified relative to some current value; this is done by preceding the multiple with either a '+' or a '–'.

6. Scoping

Layout has facilities for *scoping* of directives, i.e. for restricting their effect to a small part of the document. These scopes can be nested. At the end of the scope, the state returns to that before the start of the scope; that is, the effect of any directives invoked within the scope does not persist beyond the end of the scope.

Scoping can be done in one of two ways.

- Some directives can be followed by an opening bracket, signifying the start of the scope. The corresponding closing bracket ends the scope. The closing bracket does not imply an atom break, regardless of whether or not the directive that caused it does. Changing font, invoking a macro and some built-in directives come into this category. Four different pairs of brackets are allowed: round brackets '()', square brackets '[]', curly brackets '{}' and angle brackets '<>'.
- The directives *explicit lines*, *explicit atoms* and *explicit words* each cause a new scope lasting for the stated number of lines, atoms or words to be produced. This scope is just like any other; thus, the effect of any directives invoked within the scope doesn't persist beyond the end of the scope.

In the absence of any such scoping, the effect of the directive persists until explicitly stopped.

7. Layout Parameters

Several parameters, such as line length and the tab settings, control the layout of the formatted text. Each of these has a default value, given below. Some of the parameters, such as page length and margin sizes, are normally set no more than once within the document; others, such as the level of indentation, are likely to change frequently. The parameters can be changed with the *assign* directive. The list of parameters is given below.

- *top* (default $\frac{2}{3}$"), the top margin
- *bottom* (default $\frac{4}{3}$"), the bottom margin
- *page* (default $9\frac{2}{3}$"), the height of the page on which to allow printing. *Top* + *page* + *bottom* should always equal the physical height of the page, $11\frac{2}{3}$".
- *left* (default 0), the left margin
- *line* (default 8"), the line length
- *nls* (default $\frac{1}{8}$"), the height of each line
- *tab* (default at $\frac{2}{3}$" intervals), settings of up to 25 tab stops, numbered 1 to 25. Tab stop 0 is at the left margin and can't be changed.
- *indent* (default 0), the current level of indentation, in tab stops
- *sgap* (default $\frac{1}{6}$"), the amount of extra space inserted between sentences
- *pgap* (default $\frac{1}{4}$"), the amount of indentation at the start of each paragraph, as defined by the *new paragraph* directive
- *just* (default 0), whether or not justification is enabled (zero means disabled)
- *pageno* (default 0), the page number. If zero, then page numbering is disabled. The page number is printed in the middle of the bottom margin.
- *sectno* (default 0), the section number. If zero, or if *pageno* is zero, the section numbering is disabled. If enabled, page and section numbers are given in the form '1-2', meaning page 2 of section 1.
- *start* (default 1), the first page to be printed
- *finish* (default 9999), the last page to be printed. Together with *start*, this provides the capability of printing only a part of the document.
- *escape* (default '$'), the escape marker

8. Macros

It is possible to define directives to perform frequently required tasks. This provides some of the advantages that routines provide for programming languages; it reduces the size of the manuscript and makes it more legible.

Two different sorts of name for user-defined directives are available:

- Single-character names, for extreme brevity. The character can be any non-alphanumeric character except for the escape marker and brackets. For example, the definition

  $d / = 8r+0.3

  where font number 8 is smaller than the normal font, will allow the symbol '/' to be used for subscripts; the manuscript text

  x/(1)

  will produce the formatted text

  x_1
- Arbitrary names, consisting of a sequence of two or more letters. The definition

  $d dot = $b0 $y+0.09" $x+0.3" $x+9 $o9 $y-0.09" $x0 $t1

  produces the bullet character at the beginning of each of these paragraphs.

9. Other Uses of the Escape Marker

When the escape marker is followed by a letter or a number, it denotes a directive. Following it with any other character forces *Layout* to treat that character as normal text, regardless of any special meaning it may have. Useful cases of this are as follows:

- The escape marker followed by a space causes the space to be treated as a normal character. This can be used to prevent a line break between two words; when combined with a normal (un-escaped) space, it provides extra spacing between words.
- The escape marker followed by a full stop (or a question or exclamation mark) prevents the character from being recognised as a sentence terminator, for example when it signifies an abbreviation instead.
- Preceding a valid closing bracket, a character defined as the name of a directive or the escape marker itself with the escape marker causes that character to be included in the text as a normal character.

10. Description of Built-In Directives

There follows a description of each of the built-in directives provided by *Layout*. A table is given afterwards, summarising the directives and their various properties.

Assign ($A)

The *assign* directive is used to change the values of *Layout* parameters. For example,

```
$A just = 1
```

Several assignments can be made at once, separated by semicolons:

```
$A left = 2"; line = 4"
```

The default unit for the parameters *left*, *line*, *sgap*, *pgap* and *tab* is the width of a space in the current font; for the parameters *top*, *bottom*, *page* and *nls* it is the line height (the current value of *nls*); for the others, the values must be integers. There is a special case for the parameter *escape*; the form

```
$A escape = '%'
```

sets the escape marker to '*%*'.

The new value can be given relative to the old value, by preceding it with '+' or '–':

```
$A indent = +1
```

The old value of the parameter can be saved on a stack by replacing the '=' with '<':

```
$A indent < 1
```

saves the current value of *indent*, then sets indent to 1. A new value needn't be given:

```
$A indent <
```

saves the current value of *indent*. The saved value can be retrieved by replacing the '=' with '>':

```
$A indent >
```

retrieves the previously saved value of *indent*. It can even be retrieved and then changed:

```
$A indent > +1
```

though this feature is of dubious value.

The *assign* directive must stretch to the end of the line; no other directive or text can follow it.

There is an alternative version of this directive. The 'A' and the intervening space can be left out, but only one assignment can be made. For example:

```
$Tab = 1", 2", 3"
```

The '$A' version was going to be phased out and replaced by this alternative, to allow the name to be used for the next directive.

Explicit Atoms ($A)

The *explicit atoms* directive is used to set up a scope which lasts for a specified number of atoms.

Note the ambiguity of names between this and the *assign* directive. If the '$A' is followed directly (with no intervening spaces) by another escape marker (which is presumably the start of another directive) or a number, then it means *explicit atoms*, otherwise it means *assign*.

Blank ($B)

The *blank* directive has three uses. Firstly, it inserts blank space into the document. The amount of space is determined by its argument; the default unit is in lines, but the amount can be specified in inches too. Secondly, the directive causes a line break. The directive '$B0' thus starts a new line without inserting any blank space. Thirdly, if given a negative argument, it causes a movement up the page after the line break.

The *blank* directive has no effect if called at the top of the page, if the new page is not explicit. *New page* and *verify* are the only directives that cause an explicit new page; any other cause is implicit. The very start of the document is treated as an explicit new page.

Move to Column ($C)

This directive can be used in two ways. When used within or at the end of an atom (but not at the beginning of one), it changes the spacing between characters. It can thus be used for overprinting one character with another. The most useful way of specifying the argument in this case is relative to the current position; the default unit is the width of a space character. When used between atoms, or at the start of an atom, it acts like moving to a tab stop at the specified position.

Define ($D)

This directive is used to define macros. The name of the macro is given (either a single character or two or more letters), followed by an equals sign, and then the body of the macro. The body extends to the end of the line. The macro can be forgotten by redefining it with an empty body.

End ($E)

The *end* directive terminates the manuscript file. It is not obligatory; *Layout* will insert one itself if there is none there.

Font ($F)

This directive is used to define fonts. Its argument is the number of the font to be defined, which can be any number from 2 to 63 (fonts 0 and 1 are predefined); this is followed by an equals sign.

The definition can be given in two ways. The first way is with a font name, which can refer either to a raster font read in with the *get file* directive, or to a Times Roman font. The latter case gives access to various sizes of normal, bold, italic and scientific fonts. The names of these fonts are of the form 'TimesXnn'. 'X' must be one of 'R', 'B', 'I' and 'S', meaning 'Roman' (normal), 'Bold', 'Italic' and 'Scientific', respectively. 'nn' is the point size; available point sizes are 6, 7, 8, 9, 10, 12, 13, 14, 16, 18 and 24, with 5 point Roman, 4 and 5 point Bold and 5 point Scientific fonts also available.

The second way is to define in terms of a vector font. There are 31 Gimms fonts and 5 Times fonts available; by default they are read in, but they can be left out if they are not needed. The number of the desired font is given first. The font can be adjusted by following the definition with one of the following:

- 'S', followed by a number, gives the scale factor (magnification). 256 gives no scaling, and other numbers scale accordingly.
- 'H', followed by a size, specifies the ascender height for the font.
- 'D', followed by a size, gives a descender height for the font.
- 'W', followed by a size, makes the font fixed pitch by specifying a width for all the characters.

The last three use inches as the default unit.

Get File ($G)

This reads in a file, for example one of the files containing Times font definitions. It can also be used to read in and display a *pseudo display file*, which has been created with a graphics editor like *Edwin Draw*.

Heavy ($H)

This directive switches on and off 'emboldening' of text. This is implemented by printing characters several times, each time displaced by one pixel. The default amount is two extra pixels, but any number between 1 and 7 can be used. The directive can be followed by a bracket, restricting its scope.

Indent ($I)

The *indent* directive is an alternative to the *Layout* parameter *indent*. If the parameter has been used in the document, then the directive acts like the *tab* directive, but the argument is given relative to the current level of indentation (as opposed to the nearest tab stop). If the parameter has not been used, then the directive sets the level of indentation.

Justify ($J)

This directive causes a line break, like the '$B0' directive, but also forces justification of the completed line.

Explicit Lines ($L)

The *explicit lines* directive is used to set up a scope that lasts for the specified number of lines. If the argument is zero, then the scope will last for an indefinite number of lines; it is terminated by any line-breaking directive. Indentation is normally switched off for the extent of the scope.

An *assign* directive, calls to macros and font changes can be placed on the line after the *explicit lines* directive; they will be executed at the start of the scope, and their effect restricted to within it. There can also be several other parameters on the line. These are:

- 'M' to centre the lines
- 'I' to indent the lines
- 'H' or 'B' to make the lines bold; these can be followed by a number specifying the degree of boldness, as for the *heavy* directive
- 'U' to underline the lines; this can be followed by a number giving the depth of the underline, as for the *underline* directive

New Page ($N)

This directive closes the current page and starts a new one.

Orb ($O)

This draws a solid circle of the specified radius (in pixels) at the current position. Note that because the directive is a line-breaker, if there are no other positioning commands before it then the current position will be at the start of the line.

New Paragraph ($P)

The *new paragraph* takes as its argument a number of lines. If the current position is below the top of the page, then this number of lines is left. If there are less than two lines of space left on the page, a new page is taken. The following text is then indented by the *Layout* parameter *pgap*.

Row ($R)

This directive behaves in two different ways, depending on whether its argument is absolute or relative. The default unit is the height of the current font.

If the argument is relative, then the directive causes the vertical displacement of text on the line to be adjusted. It can take brackets to delimit its scope. In this case it doesn't cause an atom break, so it can be used within an atom.

If the argument is absolute, the directive causes a move to that position down the page, and breaks the atom. It can't be followed by brackets.

Tab ($T)

The *tab* directive causes a move to the specified tab stop. The tab stop can be given relative to the current position, in which case it causes a move by the specified number of tab positions.

Underline ($U)

This directive switches underlining on and off. Its scope can be delimited with brackets. The default distance of the underline below the base line of the text is 5 pixels, but any number from 1 up can be specified with the directive.

Verify ($V)

The *verify* directive is used to check that a certain amount of room is left on the page. If there is not, then a new page is taken. The directive can be used to ensure that a chapter of the document is kept on one page; it can also be used to check that a new section, say, is not started too close to the foot of the page.

Explicit Words ($W)

This directive is similar in effect to the *explicit atoms* directive, but using words rather than atoms.

Set X Coordinate ($X)

This directive moves to a specified position across the page. It can be followed by a comma and another position, which is treated as a *y* coordinate to move to as well. The positions can be given relative to the current position. If the pen width (see below) is not zero, then a line is drawn from the current position to the new position.

The same comment about the current position holds as for the *orb* directive.

Set Y Coordinate ($Y)

This directive is analogous to the *set x coordinate* directive, even to the extent of allowing an *x* coordinate to be specified after the *y* coordinate.

Set Pen Width ($Z)

The *set pen width* directive sets the pen width for the *set x coordinate* and *set y coordinate* directives.

Directive	Breaker[1]	Brackets allowed	Default units	Meaning
A	-	no	atoms	Explicit atoms
A	L	no	-	Assign
B	L	no	lines	Blank
C	-[2]	no	spaces	Move to Column
D	A	no	-	Define
E	L	no	-	End
F	L	no	-	Font
G	A	no	-	Get file
H	-	yes	steps	Heavy
I	L	yes	tab stops	Indent
J	L	no	-	Justify
L	L	no	lines	Explicit lines
N	L	no	-	New page
O	L	no	pixels	Orb
P	L	no	lines	New paragraph
R	-[3]	yes[4]	font height	Row
T	A	no	tab stops	Tab
U	-	yes	pixels	Underline
V	L	no	lines	Verify
W	-	no	words	Explicit words
X	L	no	pixels	Set x coordinate
Y	L	no	pixels	Set y coordinate
Z	L	no	pixels	Set pen width

1: 'A' for atom-, 'L' for line-breaker
2: Line-breaker if at start of atom
3: Atom-breaker if argument is absolute
4: Brackets not allowed if absolute

Table 1. Summary of *Layout* directives and their properties

```
$a page=10.3";top=0.5";bottom=0.5";line=6.25";left=0.8"
$a nls=2;just=1;pgap=0.3";sgap=0.2"
$a tab=0.5",0.75",1",1.25",1.5",2",2.5",3",3.5",4",4.5",5",5.5",6"
$a pageno=1
$f8=TimesR8
$f12=TimesR12
$f22=TimesB12
$f32=TimesI12
$f24=TimesB14
$f26=TimesB16
$d ^=$8$r-0.5
$d esc=$$
$d dot=$b0 $y+0.09" $x+0.3" $x+9 $o9 $y-0.09" $x0 $t1
$d section=$26
$d subsection=$p1 $v5 $t0 $24
$d unnumbered=$p1 $v5 $t0 $22
$d sdq='$c-0.3'
$d edq='$c-0.3'
$d pred=$32(PrEditor)
$d minus=-$c-0.3-
$d num=$8$r-0.6
$d denom=$8$c-1.2$r-0.5(_)$c-1.2$r+0.4
$12
$l1m
$section(Layout)

$p1 $32(Layout) is a text formatting language
defining
commands (called $32(directives)) which are embedded in
the source text (the $32(manuscript)), and which control
the way that the output (the $32(formatted text)) is produced.

$p0 This note describes $32(Layout) version 1$.5, which is an intermediate
version but the best for which the department has the source code.
The only available document on this version of $32(Layout)
('Layout 1$.5 Provisional User's Guide' by Hamish Dewar, Edinburgh University
Computer Science Department, February 1984)
is incomplete and slightly inaccurate, as well as not being widely publicised.
It is hoped that this note will serve as a manual for $32(Layout) 1$.5
and so make it easier to use.

$p0 A sample $32(Layout) 1$.5 manuscript file (the manuscript for this
document) is in the file Text:Lay.lay on the APMs.
$subsection(1. Escape Marker)

$p0 A basic characteristic of $32(Layout) is that the manuscript, in general,
represents the text to be printed unless explicitly flagged otherwise.
The method of flagging text as other than text to be printed (i$.e$. as
a directive) is to precede it with the $32(escape marker). By default, this
is '$esc', the dollar sign, but if this is an awkward choice it can be
redefined in the document.
$subsection(2. Atoms, Words and Sentences)

$p0 The $32(atom) is the basic unit of text recognised by $32(Layout). Atoms
are not split when line filling, and are not stretched when justifying the
line. An atom is defined as a sequence of characters, delimited on both sides
by one of the following:
$a indent<1
$dot a space
$dot a newline
$dot an atom breaking directive
$a indent>

$p0 A word is similar to an atom, but it consists entirely of letters; that is,
it is terminated also by any character other than a letter. It is relevant
only to the $32(explicit words) directive.

$p0 A sentence consists of a sequence of atoms starting with a capital letter
and finishing with a full stop, a question mark or an exclamation mark. The
only relevance of sentences in $32(Layout) is that extra space is inserted
between them, the actual amount of space being user-definable.
$subsection(3. Spaces and Newlines)

$p0 Within the scope of an $32(explicit lines) directive, both spaces and
newlines within the text are treated as significant, and are preserved in the
formatted text. Outside any such scope, line filling is enabled and, in general,
spaces and newlines serve only to separate words and are treated equivalently;
multiple spaces, for example, are treated as a single space. Any additional
spacing effects are produced with directives. However, some commonly occurring
requirements are covered by letting spaces and newlines be significant in
certain situations:
$a indent<1
$dot Blank lines (formed by two consecutive newlines) are preserved in the text;
thus blank space can be produced simply by leaving blank lines in the manuscript.
$dot Spaces at the start of a line of the manuscript are significant, and are
preserved in the formatted text. They also make the preceding newline
significant, that is, they force a line break. Thus, a new paragraph can be
started by leaving the required number of blank lines, then starting the
first line of the new paragraph with several spaces.
$a indent>
$b0
Spaces are also treated as significant after explicit positioning directives,
such as moving to a tab stop.
$subsection(4. Directives)

$p0 Directives can take several forms, as follows:
$a indent<1
$dot Escape marker, followed by a single letter. This invokes a basic (built
in) directive. The directive may be followed by one or more arguments, depending
on the particular directive. The case of the letter is not significant.
$dot Escape marker, followed by a number. This selects a previously defined
font. There can be up to 64 fonts, numbered from 0 to 63, of which 0 and 1
are predefined as horizontal and vertical fixed pitch (and rather grotty)
fonts.
$dot Escape marker, followed by two or more letters. This invokes a previously
defined macro. The case of the letters is again not significant.
The invocation is implemented as straight text substitution;
thus the macro itself needn't be syntactically valid as long as it becomes
valid in context, and nested macro calls are allowed (up to a maximum of
16 levels).
$dot A single non-alphanumeric symbol. This invokes a previously defined
single-character macro (the term applies to the name of the macro, rather
than its body); the invocation is implemented as above.
$a indent>

$p0 Note the possible ambiguity between a built in directive followed
by text, and a macro call. This means that some way of terminating directive
names is required. In the first three cases above, a single space after the
directive will be ignored.
$subsection(5. Arguments)

$p0 Some directives take a numeric argument, usually specifying a measurement
but sometimes specifying other items such as fonts. There is a default
unit (such as lines, pixels, tab settings and so on) for the argument, dependent
on the directive; the argument is specified as a multiple of this unit.
Where non-integer values are relevant, the multiple can be expressed as a
decimal, or as a fraction with the numerator and denominator separated by
an oblique, '$c+0.3$/$c+0.3'; the units may be changed to inches by following the argument
with '$c+0.5$"$c+0.5'.

$p0 Some directives allow the argument to be specified relative to some
current value; this is done by preceding the multiple with either a '+' or
a '$minus'.
$subsection(6. Scoping)

$p0 $32(Layout) has facilities for $32(scoping) of directives, i$.e$. for
restricting their effect to a small part of the document. These scopes can
be nested. At the end of the scope, the state returns to that before the start
of the scope; that is, the effect of any directives invoked within the scope
does not persist beyond the end of the scope.

$p0 Scoping can be done in one of two ways.
$a indent<1
$dot Some directives can be followed by an opening bracket, signifying the
start of the scope.
The corresponding closing bracket ends the scope.
The closing
bracket does not imply an atom break, regardless of whether or not the
directive that caused it does.
Changing font, invoking a macro and some built-in directives come into
this category.
Four different pairs of brackets are allowed: round brackets$ '()', square
brackets$ '[]', curly brackets$ '{}' and angle brackets$ '<>'.
$dot The directives $32(explicit lines), $32(explicit atoms) and $32(explicit
words) each cause a new scope lasting for the stated number of lines,
atoms or words to be produced. This scope is just like any other; thus, the
effect of any directives invoked within the scope doesn't persist beyond the
end of the scope.
$a indent>

$p0 In the absence of any such scoping, the effect of the directive persists
until explicitly stopped.
$subsection(7. Layout Parameters)

$p0 Several parameters, such as line length and the tab settings, control
the layout of the formatted text. Each of these has a default value, given
below. Some of the parameters, such as page length and margin sizes, are
normally set no more than once within the document; others, such as the
level of indentation, are likely to change frequently. The parameters can be
changed with the $32(assign) directive. The list of parameters is given below.
$a indent<1
$d inch=$c+0.4$r-0.2(")$c+0.4
```

$dot $32(top) (default $num(2)$denom(3)$inch), the top margin
$dot $32(bottom) (default $num(4$c+0.3)$denom(3)$inch()), the bottom margin
$dot $32(page) (default 9$num(2)$denom(3)$inch()), the height of the
page on which to allow printing. $32(Top)$ +$ $32(page)$ +$ $32(bottom) should always equal the physical hei
of the page, 11$num(2)$denom(3)$inch().
$dot $32(left) (default 0), the left margin
$dot $32(line) (default 8"), the line length
$dot $32(nls) (default $num(1)$denom(8)$inch()), the height of each line
$dot $32(tab) (default at $num(2)$denom(3)$inch() intervals), settings of up to
25 tab stops, numbered 1 to 25. Tab stop 0 is at the left margin and can't be
changed.
$dot $32(indent) (default 0), the current level of indentation, in tab stops
$dot $32(sgap) (default $num(1)$denom(6)$inch()), the amount of extra space inserted
between sentences
$dot $32(pgap) (default $num(1)$denom(4)$inch()), the amount of indentation at the
start of each paragraph, as defined by the $32(new paragraph) directive
$dot $32(just) (default 0), whether or not justification is enabled (zero means
disabled)
$dot $32(pageno) (default 0), the page number. If zero, then page numbering is
disabled. The page number is printed in the middle of the bottom margin.
$dot $32(sectno) (default 0), the section number. If zero, or if $32(pageno)
is zero, the section numbering is disabled. If enabled, page and section
numbers are given in the form '1-2', meaning page 2 of section 1.
$dot $32(start) (default 1), the first page to be printed
$dot $32(finish) (default 9999), the last page to be printed.
Together with $32(start), this provides the capability of printing only a
part of the document.
$dot $32(escape) (default '$esc()'), the escape marker
$a indent>
$subsection(8. Macros)

$p0 It is possible to define directives to perform frequently required tasks.
This provides some of the advantages that routines provide for programming
languages; it reduces the size of the manuscript and makes it more legible.

$p0 Two different sorts of name for user-defined directives are available:
$a indent<1
$dot Single-character names, for extreme brevity. The character can be any
non-alphanumeric character except for the escape marker and brackets.
For example, the definition
$b0 $t+1 $esc()d / = $esc()8$esc()r+0.3
$b0 where font number 8 is smaller than the normal font, will allow the
symbol '/' to be used for subscripts; the manuscript text
$b0 $t+1 x/(1)
$b0 will produce the formatted text
$b0 $t+1 x$8($r+0.3(1))
$dot Arbitrary names, consisting of a sequence of two or more letters. The
definition
$b0 $t+1 $esc()d dot = $esc()b0 $esc()y+0.09" $esc()x+0.3" $esc()x+9 $esc()o9 $esc()y-0.09" $esc()x0 $esc()t
$b0 produces the bullet character at the beginning of each of these
paragraphs.
$a indent>
$subsection(9. Other Uses of the Escape Marker)

$p0 When the escape marker is followed by a letter or a number, it denotes
a directive. Following it with any other character forces $32(Layout) to treat
that character as normal text, regardless of any special meaning it may have.
Useful cases of this are as follows:
$a indent<1
$dot The escape marker followed by a space causes the space to be treated as
a normal character. This can be used to prevent a line break between two words;
when combined with a normal (un-escaped) space,
it provides extra spacing between words.
$dot The escape marker followed by a full stop (or a question or exclamation
mark) prevents the character from being recognised as a sentence terminator, for
example when it signifies an abbreviation instead.
$dot Preceding a valid closing bracket, a character defined as the name of a
directive or the escape marker itself with the escape marker causes that
character to be included in the text as a normal character.
$a indent>
$subsection(10. Description of Built-In Directives)

$p0 There follows a description of each of the built-in directives provided
by $32(Layout). A table is given afterwards, summarising the directives and
their various properties.
$unnumbered(Assign ($esc()A))

$p0 The $32(assign) directive is used to change the values of $32(Layout)
parameters. For example,
$b0 $t+1 $esc()A just = 1
$b0 Several assignments can be made at once, separated by semicolons:
$b0 $t+1 $esc()A left = 2"; line = 4"

$p0 The default unit for the parameters $32(left), $32(line), $32(sgap),
$32(pgap) and $32(tab) is the width of a space in the current font; for the
parameters $32(top), $32(bottom), $32(page) and $32(nls) it is the line height
(the current value of $32(nls)); for the others, the values must be integers.
There is a special case for the parameter $32(escape); the form
$b0 $t+1 $esc()A escape = '%'
$b0 sets the escape marker to '%'.

$p0 The new value can be given relative to the old value, by preceding it
with '+' or '$minus':
$b0 $t+1 $esc()A indent = +1

$p0 The old value of the parameter can be saved on a stack by replacing the
'=' with '<':
$b0 $t+1 $esc()A indent < 1
$b0 saves the current value of $32(indent), then sets indent to 1. A new value
needn't be given:
$b0 $t+1 $esc()A indent <
$b0 saves the current value of $32(indent). The saved value can be retrieved
by replacing the '=' with '>':
$b0 $t+1 $esc()A indent >
$b0 retrieves the previously saved value of $32(indent). It can even
be retrieved and then changed:
$b0 $t+1 $esc()A indent > +1
$b0 though this feature is of dubious value.

$p0 The $32(assign) directive must stretch to the end of the line; no other
directive or text can follow it.

$p0 There is an alternative version of this directive. The 'A' and the
intervening space can be left out, but only one assignment can be made. For
example:
$b0 $t+1 $esc()Tab = 1", 2", 3"
$b0 The '$esc()A' version was going to be phased out and replaced by this alternative,
to allow the name to be used for the next directive.
$unnumbered(Explicit Atoms ($esc()A))

$p0 The $32(explicit atoms) directive is used to set up a scope which lasts
for a specified number of atoms.

$p0 Note the ambiguity of names between this and the $32(assign) directive.
If the '$esc()A' is followed directly (with no intervening spaces) by another
escape marker (which is presumably the start of another directive) or a number,
then it means $32(explicit atoms), otherwise it means $32(assign).
$unnumbered(Blank ($esc()B))

$p0 The $32(blank) directive has three uses. Firstly, it inserts blank space
into the document. The amount of space is determined by its argument; the
default unit is in lines, but the amount can be specified in inches too.
Secondly, the directive causes a line break. The directive '$esc()B0' thus
starts a new line without inserting any blank space. Thirdly, if given a
negative argument, it causes a movement up the page after the line break.

$p0 The $32(blank) directive has no effect if called at the top of the page,
if the new page is not explicit. $32(New page) and $32(verify)
are the only directives that cause an explicit new page; any other cause
is implicit. The very start of the document is treated as an explicit new page.
$unnumbered(Move to Column ($esc()C))

$p0 This directive can be used in two ways. When used within or at the end of
an atom (but not at the beginning of one), it changes the spacing between
characters. It can thus be used for overprinting one character with another.
The most useful way of specifying the argument in this case is relative to
the current position; the default unit is the width of a space character. When
used between atoms, or at the start of an atom, it acts like moving to a
tab stop at the specified position.
$unnumbered(Define ($esc()D))

$p0 This directive is used to define macros. The name of the macro is given
(either a single character or two or more letters), followed by an equals sign,
and then the body of the macro. The body extends to the end of the line. The
macro can be forgotten by redefining it with an empty body.
$unnumbered(End ($esc()E))

$p0 The $32(end) directive terminates the manuscript file. It is not
obligatory; $32(Layout) will insert one itself if there is none there.
$unnumbered(Font ($esc()F))

$p0 This directive is used to define fonts. Its argument is the number of the
font to be defined, which can be any number from 2 to 63 (fonts 0 and 1 are
predefined); this is followed by an equals sign.

$p0 The definition can be given in two ways. The first way is with a font name,
which can refer either to a raster font read in with the $32(get file)
directive, or to a Times Roman font. The latter case gives access to various
sizes of normal, bold, italic and scientific fonts. The names of these fonts
are of the form 'TimesXnn'. 'X' must be one of 'R', 'B', 'I' and 'S', meaning
'Roman' (normal), 'Bold', 'Italic' and 'Scientific', respectively. 'nn' is the
point size; available point sizes are 6, 7, 8, 9, 10, 12, 13, 14, 16, 18 and

24, with 5 point Roman, 4 and 5 point Bold and 5 point Scientific fonts also available.

$p0 The second way is to define in terms of a vector font. There are 31 Gimms fonts and 5 Times fonts available; by default they are read in, but they can be left out if they are not needed. The number of the desired font is given first. The font can be adjusted by following the definition with one of the following:
$a indent<1
$dot 'S', followed by a number, gives the scale factor (magnification). 256 gives no scaling, and other numbers scale accordingly.
$dot 'H', followed by a size, specifies the ascender height for the font.
$dot 'D', followed by a size, gives a descender height for the font.
$dot 'W', followed by a size, makes the font fixed pitch by specifying a width for all the characters.
$a indent>
The last three use inches as the default unit.
$unnumbered(Get File ($esc()G))

$p0 This reads in a file, for example one of the files containing Times font definitions. It can also be used to read in and display a $32(pseudo display file), which has been created with a graphics editor like $32(Edwin Draw).
$unnumbered(Heavy ($esc()H))

$p0 This directive switches on and off 'emboldening' of text.
This is implemented by printing characters several times, each time displaced by one pixel. The default amount is two extra pixels, but any number between 1 and 7 can be used.
The directive can be followed by a bracket, restricting its scope.
$unnumbered(Indent ($esc()I))

$p0 The $32(indent) directive is an alternative to the $32(Layout) parameter $32(indent). If the parameter has been used in the document, then the directive acts like the $32(tab) directive, but the argument is given relative to the current level of indentation (as opposed to the nearest tab stop). If the parameter has not been used, then the directive sets the level of indentation.
$unnumbered(Justify ($esc()J))

$p0 This directive causes a line break, like the '$esc()B0' directive, but also forces justification of the completed line.
$unnumbered(Explicit Lines ($esc()L))

$p0 The $32(explicit lines) directive is used to set up a scope that lasts for the specified number of lines. If the argument is zero, then the scope will last for an indefinite number of lines; it is terminated by any line-breaking directive. Indentation is normally switched off for the extent of the scope.

$p0 An $32(assign) directive, calls to macros and font changes can be placed on the line after the $32(explicit lines) directive; they will be executed at the start of the scope, and their effect restricted to within it. There can also be several other parameters on the line. These are:
$a indent<1
$dot 'M' to centre the lines
$dot 'I' to indent the lines
$dot 'H' or 'B' to make the lines bold; these can be followed by a number specifying the degree of boldness, as for the $32(heavy) directive
$dot 'U' to underline the lines; this can be followed by a number giving the depth of the underline, as for the $32(underline) directive
$a indent>
$unnumbered(New Page ($esc()N))

$p0 This directive closes the current page and starts a new one.
$unnumbered(Orb ($esc()O))

$p0 This draws a solid circle of the specified radius (in pixels) at the current position. Note that because the directive is a line-breaker, if there are no other positioning commands before it then the current position will be at the start of the line.
$unnumbered(New Paragraph ($esc()P))

$p0 The $32(new paragraph) takes as its argument a number of lines. If the current position is below the top of the page, then this number of lines is left. If there are less than two lines of space left on the page, a new page is taken. The following text is then indented by the $32(Layout) parameter $32(pgap).
$unnumbered(Row ($esc()R))

$p0 This directive behaves in two different ways, depending on whether its argument is absolute or relative.
The default unit is the height of the current font.

$p0 If the argument is relative, then the directive causes the vertical displacement of text on the line to be adjusted. It can take brackets to delimit its scope. In this case it doesn't cause an atom break, so it can be used within an atom.

$p0 If the argument is absolute, the directive causes a move to that position down the page, and breaks the atom. It can't be followed by brackets.
$unnumbered(Tab ($esc()T))

$p0 The $32(tab) directive causes a move to the specified tab stop. The tab stop can be given relative to the current position, in which case it causes a move by the specified number of tab positions.
$unnumbered(Underline ($esc()U))

$p0 This directive switches underlining on and off. Its scope can be delimited with brackets. The default distance of the underline below the base line of the text is 5 pixels, but any number from 1 up can be specified with the directive.
$unnumbered(Verify ($esc()V))

$p0 The $32(verify) directive is used to check that a certain amount of room is left on the page. If there is not, then a new page is taken. The directive can be used to ensure that a chapter of the document is kept on one page; it can also be used to check that a new section, say, is not started too close to the foot of the page.
$unnumbered(Explicit Words ($esc()W))

$p0 This directive is similar in effect to the $32(explicit atoms) directive, but using words rather than atoms.
$unnumbered(Set X Coordinate ($esc()X))

$p0 This directive moves to a specified position across the page. It can be followed by a comma and another position, which is treated as a $32(y) coordinate to move to as well. The positions can be given relative to the current position. If the pen width (see below) is not zero, then a line is drawn from the current position to the new position.

$p0 The same comment about the current position holds as for the $32(orb) directive.
$unnumbered(Set Y Coordinate ($esc()Y))

$p0 This directive is analogous to the $32(set x coordinate) directive, even to the extent of allowing an $32(x) coordinate to be specified after the $32(y) coordinate.
$unnumbered(Set Pen Width ($esc()Z))

$p0 The $32(set pen width) directive sets the pen width for the $32(set x coordinate) and $32(set y coordinate) directives.

$t12 Jeremy Gibbons
$t12 July 1987
$v32
$a line<1"
$l0m

Directive

A
A
B
C
D
E
F
G
H
I
J
L
N
O
P
R
T
U
V
W
X
Y
Z
$b-26
$a left<+1"
$l0m
$r+0.3($r-2/300"(Breaker^(1)))

-
L
L
-^(2)
A
L
L
A
-
L
L
L
L
L
L
-^(3)
A
-
L
-
L
L
L
$b-26
$a left=+1"
$10m
Brackets
allowed

no
no
no
no
no
no
no
no
yes
yes
no
no
no
no
no
yes^(4)
no
yes
no
no
no
no
no
$b-26
$a left=+1";line<1.5"
$10m
Default
units

atoms
-
lines
spaces
-
-
-
-
steps
tab stops
-
lines
-
pixels
lines
font height
tab stops
pixels
lines
words
pixels
pixels
pixels
$b-26
$a left=+1.7";line>
$10

Meaning

Explicit atoms
Assign
Blank
Move to Column
Define
End
Font
Get file
Heavy
Indent
Justify
Explicit lines
New page
Orb
New paragraph
Row
Tab
Underline
Verify
Explicit words
Set x coordinate
Set y coordinate
Set pen width
$a left>; line>

$15

$t4 1: 'A' for atom-, 'L' for line-breaker
$t4 2: Line-breaker if at start of atom
$t4 3: Atom-breaker if argument is absolute
$t4 4: Brackets not allowed if absolute
$13m

Table 1. Summary of $32(Layout) directives and their properties
$b-8
$y-1/8"
$z2 $y-6.875" $x+6.2" $y+6.875" $x-6.2"
$z0 $y-6.25" $z2 $x+6.2" $z0
$x-5.2" $y-0.625"
$z2 $y+6.875" $z0 $x+1"
$z2 $y-6.875" $z0 $x+1"
$z2 $y+6.875" $z0 $x+1.5"
$z2 $y-6.875" $z0

Font TimesR5

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR6

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR7

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR8

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR9

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR10

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR12

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR13

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR14

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR16

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR18

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesR24

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

Font TimesI6

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesI7

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesI8

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesI9

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesI10

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesI12

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesI13

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesI14

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesI16

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesI18

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesI24

! $%&'() ,-./0123456789:; ?*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdef ghijklmnopqrstuvwxyz

Font TimesB4

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB5

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB6

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB7

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB8

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB9

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB10

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB12

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB13

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB14

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

Font TimesB16

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

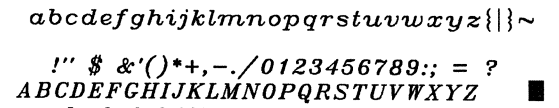
Font TimesB18

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

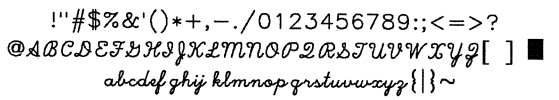
Font TimesB24

! $%&'()* ,-./0123456789:; ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
'abcdefghijklmnopqrstuvwxyz

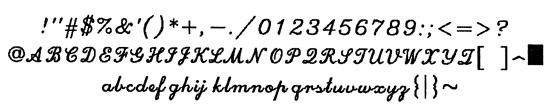
Gimms font 26



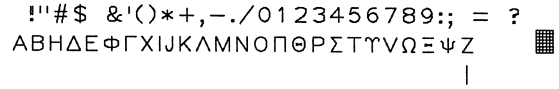
Gimms font 32



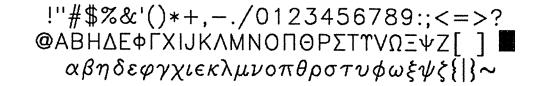
Gimms font 33



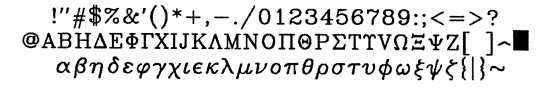
Gimms font 41



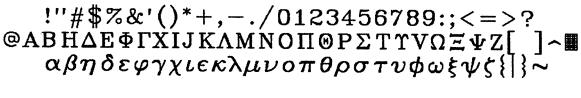
Gimms font 42



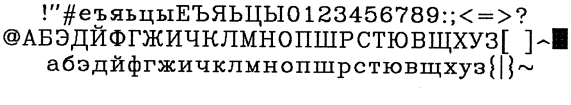
Gimms font 43



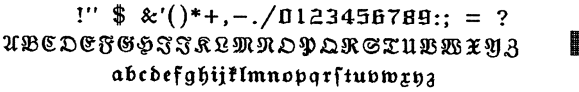
Gimms font 44



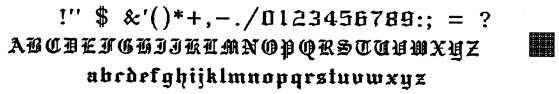
Gimms font 48



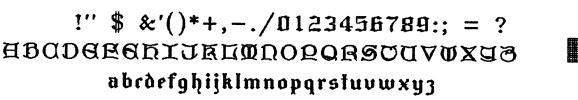
Gimms font 51



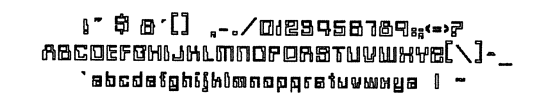
Gimms font 52



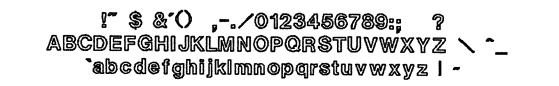
Gimms font 53



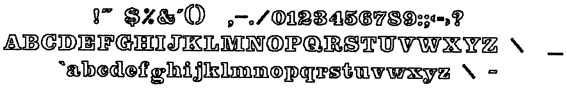
Gimms font 60



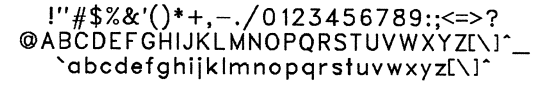
Gimms font 61



Gimms font 62

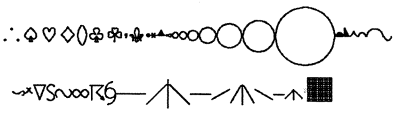


Gimms font 63

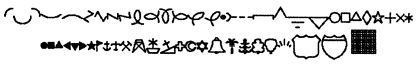


Gimms font 64

Gimms font 2



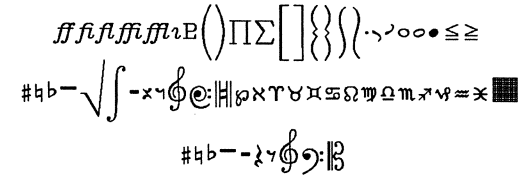
Gimms font 3



Gimms font 4

ffﬁﬂﬃﬄıϵθϕς‖±∓∝⌢ˆˇ′″‴⊂∪⊃∩∈⟨≡⟩→
↑←↓∂∇√∫∮∞§†‡∃⊙♀♀⊕♂♃♄♁Ψℭ♂∗ΩʊⅢ

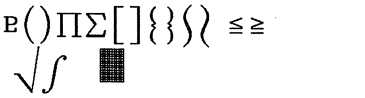
Gimms font 5



Gimms font 6

ffﬁﬂﬃﬄıϵθϕς‖±∓∝⌢ˆˇ′″‴⊂∪⊃∩∈⟨≡⟩→
↑←↓∂∇√∫∮∞§†‡∃⊙♀♀⊕♂♃♄♁Ψℭ♂∗ΩʊⅢ

Gimms font 7



Gimms font 11

!"#$ &'()*+,-./0123456789:; = ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Gimms font 12

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]
abcdefghijklmnopqrstuvwxyz{|}~

Gimms font 13

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]^
abcdefghijklmnopqrstuvwxyz{|}~

Gimms font 14

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]^
abcdefghijklmnopqrstuvwxyz{|}~

Gimms font 15

!" $ &'()*+,-./0123456789:; = ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Gimms font 16

!" $ &'()*+,-./0123456789:; = ?
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Gimms font 17

!"#$%&'()*+,-./0123456789:;<=>?
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Gimms font 23

!"#$%&'()*+,-./0123456789:;<=>?
@*ABCDEFGHIJKLMNOPQRSTUVWXYZ*[ ]^
abcdefghijklmnopqrstuvwxyz{|}~

Gimms font 24

!"#$%&'()*+,-./0123456789:;<=>?
@*ABCDEFGHIJKLMNOPQRSTUVWXYZ*[ ]^