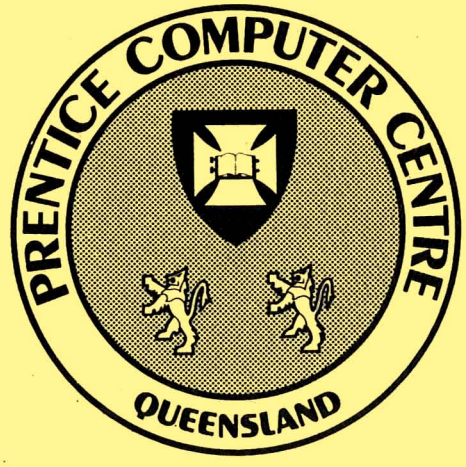
PRENTICE COMPUTER CENTRE

University of Queensland

Technical Manual No. 7

TYPESETTING MANUAL



MNT-7
June 1983

PRENTICE COMPUTER CENTRE
UNIVERSITY OF QUEENSLAND

Typesetting Manual

ANDREW BROUGHTON
AND
BARRY MAHER

JUNE 1983

This manual has been authorized by the Director of the Prentice Computer Centre.

Typeset at the Prentice Computer Centre, University of Queensland.

Printed at The Printery, University of Queensland.

Contents

Introduction

This manual has two aspects — the introduction of the ideas and conventions associated with common typesetting practice to users with no previous experience in this area; and the detailing of the commands available in the typesetting program JUSTIF, together with demonstrations of some of the more common applications of these commands.

The first aspect is the more difficult to achieve, since it requires attention to a variety of ideas which have developed over centuries of practice and constitutes one part of the training undertaken by a member of the printing profession. Since it depends so much on conventions, personal preferences and "in-house" rules (which may vary from one institution to another), this is clearly a task which is both large and ill-defined. In an attempt to make the transition to typesetting manageable for the new user, yet enable the production of output with a professional appearance, Part A of this manual is designed for the beginner who requires some idea of the ideas and scope of typesetting by demonstrating familiar tasks to which the user can relate ... it is a gentle introduction.

Once these basic ideas are known (either from previous knowledge or from studying Part A), more advanced techniques can be developed via examination of Part B, which reveals and demonstrates the complete list of typesetting commands available in the current package. Users with this prior knowledge will require Part B only, and then as a reference manual only.

Part C is devoted to the procedural aspects associated with the generation and processing of files containing typesetting commands, and is an important element for all users.

The Appendix section contains details pertaining to available characters, available typefaces, the accessing of various characters in various styles, etc. Again, all users will have need of this section for reference purposes.

New users who will have a responsibility not only for knowing the relevant command set, but also for the ultimate design and production of documents (reports, magazines, etc.) should also endeavour to obtain access to a publication which provides details of common conventions, order of book-makeup, etc. A suitable publication of this sort would be *A Manual of Style*, published by the University of Chicago Press. In addition, useful ideas and practices can be learned all the time simply by looking at a range of printed material of the same category as the material you are attempting to produce.

PART A

BEGINNER'S GUIDE TO COMPUTERISED PHOTOTYPESETTING AT THE PRENTICE COMPUTER CENTRE

CHAPTER A1
Introduction and Basic Concepts

The purpose of this sub-document is to provide a simple practical introduction to the use of the ITPS typesetting package available at the Prentice Computer Centre. It is directed at potential users with little or no previous knowledge of typesetting, giving initial instructions in and demonstrations of the terms and basic techniques. It does, however, assume familiarity with fundamental ideas such as

- how to log on to the UQKL-10 system
- how to operate a normal general-purpose computer terminal
- how to perform basic operating system commands (e.g. DIRECT, DELETE, TYPE, PRINT, etc.)
- how to use any one of the available editing programs (UQEDIT or SED or TECO or SOS or VIDED or VISED) to generate and modify files
- how to log off the system at the end of a terminal session.

Such topics are treated in appropriate detail in other manuals (such as MNT-2, MNT-6, etc) and documents, and intending users without this knowledge should refer to such documentation and/or attend relevant courses conducted by the Prentice Computer Centre before proceeding in detail with the current manual.

WHAT IS TYPESETTING?

The composition (or setting) of text involves positioning characters into organized "units" (such as lines, paragraphs and pages) in accordance with certain pre-determined specifications designed to display the text to maximum advantage.

To the extent that the comment above could also be said to apply to, say, typed material, it should be pointed out immediately that:

(a) typesetting generally allows a greater variety and convenient flexibility of *style* of type (character shape) and *size* of type, a small sample of which is displayed below:

English Times 10 point
Times Italic 9 point
Times Bold 14 point
Helios 8 point
Souvenir Bold 12 point
University 11 point

(b) normally, one has access to a greater *range of characters* in any given style and size. In addition to the "normal" alphabetic and numeric characters and punctuation symbols, one commonly finds "additional" characters, e.g. "highlight" characters such as •, ★; "combination-characters" such as ff, fl, ffi (called ligatures); and a greater variety of "dashes" including - (hyphen), – (en-dash) and — (em-dash), all of which add a sense of style and variety to the typeset article.

(c) the conventions adopted by typists and typesetters are frequently *very different*. Generally, this is because the typesetter has ready access to a greater variety of styles, sizes and special characters, and finer control over the device producing the final product, and so the standards have developed in the certainty of the availability of such discriminators. The typewriter is limited as a compromise between handwriting and typesetting.

The net effect is to produce a final product which should be visually more pleasing and of a higher quality than a corresponding typed result. The reason is simply that typesetting machines are much more sophisticated in operation and are capable of greater flexibility and control.

Thus, the user of the typesetting facilities available should endeavour, as far as possible, to adopt and conform to the standards of the typesetting art.

Two final points should be stressed:

1. In typesetting, normally any given pair of characters will have *different widths*, so that techniques commonly used for vertical alignment in, say, typing will not necessarily be successful in typesetting (e.g. "counting characters"). Thus, the new user may have to learn some different strategies to achieve desired results.
2. With computerised phototypesetting, the final appearance of the document being prepared is never evident until the product is actually produced ... you work "blind" (whereas, with a typewriter, you can see, for example, where one page finishes and another begins as the text is being entered). In this respect, there is no substitute for experience and confidence ... practice in the various techniques is essential.

AN OVERVIEW OF THE PROCESSES

ITPS-10 is a typesetting package available to all users of the UQKL-10 system, which gives the user control, via an extensive range of commands, over the final appearance of text material produced as camera-ready copy from either the COMPUGRAPHIC 8400 typesetting machine located at the Prentice Computer Centre, or an APS-5 machine located at Queensland Newspapers.

To achieve the end-result, there are just three essential steps:

(a) the creation of a file (called the source file) which consists of a mixture of the text required to appear in the final product and the commands which are interpreted by the ITPS-10 package to style (or format) that text;

A typical source file (XYZ.TYP) may appear:

```
[*cg8400][f100][p10][v11][c16.0]
This is a simply-styled document which is set up as a file containing
both text and commands./l
The number of commands required will typically depend on the
complexity of setting desired in the final product./l
Variations in [f101]style[f100] are usually
accomplished fairly easily./l
```

(b) the processing of the file created in (a) by the ITPS-10 program called JUSTIF, which performs all the necessary formatting, including the two basic operations of *hyphenation* and *justification*, in accordance with certain in-built rules and conventions and by the commands entered by the user in the source file, producing a second file used below in stage (c);

The procedure to follow is:

```
. r justif
[JUSTIF Version 301 (160400)-2]
* XYZ
[JNHQUE - Justifying file DSK:XYZ.TYP]
* ^C
```

(c) the transmission of the file generated by JUSTIF in (b) to the COMPUGRAPHIC 8400 phototypesetter (or, occasionally, some other appropriate device), producing the final product, which has the form of the text displayed on chemically-treated paper ready for delivery to a printing establishment.

For transmission:

```
. /cg XYZ.LST
[LPT026:XYZ/Seq:173554/LIMIT:15, 1 File]
```

Final product appears:

This is a simply-styled document which is set up as a file containing both text and commands.
The number of commands required will typically depend on the complexity of setting desired in the final product.
Variations in *style* are usually accomplished fairly easily.

In addition to these essential operations, there are a number of optional (but often desirable, and frequently used) additional steps available. These include:

- editing the source file to amend either text or command items
- obtaining less expensive, simulated copies for the purposes of proof-reading and general inspection of style
- general "house-keeping" functions, such as deleting unwanted versions of documents, archiving files for updating at a much later time, etc.

After a brief description of the form of ITPS commands in the next chapter, you will be able to try setting a few simple standard styles for yourself.

CHAPTER A2
ITPS Commands and Codes

As indicated in the previous chapter, the commands required by ITPS are inserted within the body of text in the source file. These commands control the format of the finished product, allowing the user to select and change such items as the style of type to be used, the size of type to be used, the spacing between lines, any desired indents, the general style of paragraphs, the width of type on a page, the depth of pages, etc.

Most ITPS commands take one of the following general forms:

[command-code]
or **[command-code** *value(s)***]**
or **/command-code**

Generally, these command-codes are just one or two characters long and are mnemonic in form—I for indent, P for pointsize, L for left, R for right, etc. These command-codes may be entered using either upper-case or lower-case—ITPS will accept both forms.

Reference to the above forms shows that the characters [and] and / are "reserved", i.e. they have special significance to the typesetting programs. Certain other characters, viz. + and @, also fall into this category. So, if we require such characters to actually appear in the final document, they have to be accessed in a special way. In particular, we use

/[in the source file to obtain the printing character [in the output document,
/] in the source file to obtain the printing character] in the output document,
and // in the source file to obtain the printing character / in the output document.

In this manual, we will refer to the combination-symbols /[, /] and // as the respective *access codes* for the characters [,] and /.

ACCESS CODES

It should be noted that different "styles" have different sets of characters available (See Appendix E),
e.g. in the style referred to as English Times, we may call up characters such as ✓, ★, @, ◄, none of which is available in the style called Baskerville; but Baskerville includes the characters £, ⅝, ffi, and accents (e.g. ^, ~, etc), which English Times does not.

In the system implemented at the Prentice Computer Centre, the following characters are available in *all* "text" styles:

- all alphabetic characters—both upper- and lower-case
- all numeric characters—0 through 9
- common punctuation marks—? ! . , : ; etc

i.e. about 77 characters in all.

Each style, or *font*, contains 118 characters, so the balance of about 41 characters may differ from font to font.

Each of these "miscellaneous" characters are called up by an "access code" such as /M or /N, or +B or +S, etc. (Note that the character / has a dual role—it is used with both command codes and access codes, whereas + is used only in access codes.) A complete list of these access codes for each individual font is displayed in the "font-blueprint" of Appendix E.

Notice that the same access code can produce a different character, depending on the font currently in use, e.g. (from Appendix E) the code +b is related to the character ● if the font is English Times, whereas the same code is associated with the character ☙ for Tiffany.

Obviously, then, when you get down to the serious business of setting type, the information of Appendix E will be indispensable, both as a reference for the availability of characters in given fonts and as a reference for the appropriate access code when the character exists in such a font.

CHAPTER A3
Some Typical Examples

Preliminary Notes:

1. Users new to typesetting are encouraged to perform some (at least) of the examples shown in this chapter.
2. Some of the terms (e.g. pica, point, etc) may be foreign to you. These will be explained in working terms as they are encountered.

EXAMPLE 1

Introduction: This example demonstrates that, to obtain output from a text file, it is strictly necessary only to insert one initial command into that text file, since any other essential commands required for formatting will adopt values built into the system if they are not supplied by the user.

Also, whereas it shows the editor UQEDIT being used to establish the file and also to make subsequent amendments to that file, if you are more familiar with some other editing program, then feel free to use that alternative editor in your own practice.

Step 1:

```
. create S1.TYP
Input:
[*cg8400]
This short file is to be
used to produce typeset
output using the faculties
of the Prentice Computer Centre.
Since we don't know any commands
at this stage, we won't use any!
As commands are introduced, they
will be tried.

* file
.
```

Explanatory Notes

(a) Always give the source file the extension .TYP
(b) The command [*cg8400] (or [*aps]) *must* always be inserted as the *first line* of the source file. It nominates the device to be used to produce the final product.
(c) Note the literal error in the fourth input line ('faculties' should read 'facilities').
(d) Note that text is *freely input*, with no attempt at arrangement. *But*, don't split words over two input lines.

Step 2:

```
. r justif
[JUSTIF Version 301 (160400)-2]
* S1
[JNHQUE - Justifying file DSK:S1.TYP]
[JNHFIN - S1 DONE]
* ^C
.
```

At this stage, a new file S1.LST has been produced
Return to monitor level

The file S1.LST produced above is a file in which all the commands have been translated into instructions which the phototypesetting machine can interpret, i.e. it is ready to send to the output device.

Step 3:

```
. /cg s1.lst
[LPT026:S1 = /Seq:175161/LIMIT:23, 1 File]
```

Note that the file S1.LST will disappear from your area.

This file has now been queued for processing by the COMPUGRAPHIC 8400 typesetting machine. Suffice it to say at this stage that this machine is operated so that, for the cheapest rate, the above command guarantees overnight turnaround but, if demand is sufficiently heavy, output may be obtained more quickly than that.

When you collect your output, it should appear as:

This short file is to be used to produce typeset output using the faculties of the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any! As commands are introduced, they will be tried.

Note the spelling mistake is still present, but the output text is justified.

If we are not satisfied with the result, either because of textual errors (as in this case) or because of general appearance, we return to the source file S1.TYP, make the necessary amendments using an editing program, re-process the revised version with the JUSTIF program, and re-submit the resultant S1.LST file to the phototypesetting machine, as indicated below:

Step 4:

```
. edit S1.TYP
[*cg8400]
* locate /faculties/
output using the faculties
* change /facul/facili/
output using the facilities
* file
[EDIFIL: S1.TYP]
.
```

Step 5:

```
. r justif
[JUSTIF Version 301 (160400)-2]
* S1
[JNHQUE - Justifying file DSK:S1.TYP]
[JNHFIN - S1 DONE]
* ^C
.
```

Step 6:

```
. /cg S1.LST
[LPT026:S1 = /Seq:175167/LIMIT:6, 1 File]
.
```

This time the output should appear:

This short file is to be used to produce typeset output using the facilities of the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any! As commands are introduced, they will be tried.

Notice how the alteration of the word "faculties" into "facilities" has completely altered the first line set. The whole of the word "Computer" has been moved to the second line and, as a consequence, the word "command" previously appearing at the end of the second line is now hyphenated. It is not uncommon in typesetting for a "minor" change in input to produce a "major" effect on the output.

Note: In all subsequent examples, only the content of the .TYP file and the appearance of the final product will be shown.

EXAMPLES 2 – CHANGING TYPEFACE

Introduction: In this series of examples, we will demonstrate how the final appearance of a document can be altered by the selection of different *typefaces*.

Over time, many different "styles" of type have been designed to suit different purposes—the advertiser requires explosive, eye-catching styles, whereas an academic text is generally better suited by a more reserved appearance. These collections of distinctive representations of alphabetic and other characters are called typefaces. Organizationally, characters belonging to the same typeface are gathered into a collection, called a *font*. With experience, you will learn to recognize some of these different fonts—all the available fonts on this system are displayed in Appendix A, Appendix B and Appendix E.

For purposes of identification, each font is associated with a unique number, as given in Appendix A—English Times is font 100, Baskerville is font 110, Helios is font 520, etc.

In Example 1, no font was specified in the source file; in this case, *font 100*, i.e. *English Times*, is automatically chosen by the system.

In Examples 2(a)-2(f) below, a variety of different fonts will be applied to text similar to that used in Example 1.

The form of the relevant command is **[f*?*]**, where *?* is the font-number of the desired typeface.

(**f** denotes "font")

Example 2(a)

Source file contains:

```
[*cg8400]
[f120]                          Garth Graphic face
This short file is to be
...
at this stage, so we won't use any!
```

Output appears:

This short file is to be used to produce typeset output using the facilities of the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any!

Example 2(b)

Source file contains:

```
[*cg8400]
[f101]                          Times Italic face
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities of the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any!

Example 2(c)

Source file contains:

```
[*cg8400]
[f520]                          Helios face
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities at the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any!

Example 2(d)

Source file contains:

```
[*cg8400]
[f200]                          Korinna Extrabold face
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities at the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any!

Example 2(e)

```
[*cg8400]
[f280]                          Florentine Script face
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities at the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any!

Example 2(f)

Source file contains:

```
[*cg8400]
[f140]                          Mallard face
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities at the Prentice Computer Centre. Since we don't know any commands at this stage, we won't use any!

Moreover, a variety of fonts may be used within a single document for reasons of highlighting particular phrases, headings, etc.

Example 2(g)

Source file contains:

```
[*cg8400]
[f150]                          Start with Souvenir
For purposes of emphasis, it may
be desirable to have [f153]some     Use Souvenir Bold
phrases in bold type[f150] and      Restore original face
also [f151]some others in italic, or    Swap to Souvenir Italic
sloping, type[f150] and perhaps even    Restore again
some phrases [f153]in bold italic[f150], Use Bold Italic face temporarily
all in the same document.
```

Output appears:

For purposes of emphasis, it may be desirable to have some phrases in bold type and also *some others in italic, or sloping, type* and perhaps even some phrases *in bold italic*, all in the same document.

Notes on Style:

1. Typeface changes should be used consistently and sensibly. Too much bold, for example, can reduce the visual importance of key phrases.
2. In general, the various typefaces are chosen from the same 'family', i.e. if the regular face is, say, Baskerville and we require some bold text, then we choose Baskerville Bold (not Times Bold, nor Univers Bold, etc).

EXAMPLES 3 – CHANGING WIDTH

Introduction: Another general characteristic which will considerably alter the final appearance is the *measure*, or width across the page, to which the type is set.

This measure is always quoted in units called *picas* and *points*,

where 6 picas = 1 inch
and 12 points = 1 pica.

In this context, the notation 4.6 indicates 4 picas and 6 points (i.e. 4½ picas);
likewise 19.11 indicates 19 picas and 11 points
(i.e. decimal notation is *not* used).

In all of our examples thus far, we have not specified a measure, so the "system standard" has been applied, viz. 36.0 picas (i.e. 6 inches).

The form of the command to alter width of setting is **[c***?.?***]**, where *?.?* denotes the required measure in the picas.points notation.

(**c** denotes "column measure")

Example 3(a)

Source file contains:

```
[*cg8400]
[c24.0]
This short file is to be
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities at the Prentice Computer Centre. Since we don't know many commands at this stage, we won't use any!

Example 3(b)

Source file contains:

```
[*cg8400]
[c12.0]
...
```

Output appears:

This short file is to be used to produce typeset output using the facilities at the Prentice Computer Centre. Since we don't know many commands at this stage, we won't use any!

Note: As column measure is reduced, interword-spacing introduced during the justification process may become more evident, and hyphenation may have to be used more extensively.

EXAMPLE 4 — CHANGING POINTSIZE (SIZE OF TYPE)

Introduction: Variation in *pointsize*, i.e. the height of type used, also has a marked impact on the appearance of the final product, as the examples below show.

Pointsize, as used in printing, is not a precise term, but varies, for example, considerably from one typeface to another. However, it is expressed in the unit of the point. If the COMPUGRAPHIC 8400 system is used, the related software allows the use of 30 distinct type sizes, ranging from 5-point type (This is 5-point type in Baskerville Medium) to 72-point type

(VERY BIG!)

In all the previous examples, no pointsize was specified, so output is automatically set to 10-point.

The form of the command to change current pointsize is **[p*?*]**, where *?* is the pointsize desired.

(**p** denotes "pointsize")

Example 4(a)

Source file contains:

```
[*cg8400]
[c16.0]
[p8]
This sample is set in English Times 8-point. There are 30 sizes allowed, from 5 point
to 72 point. The width of the column measure to be used often determines the most
appropriate size of type to use. This manual is set primarily in 11 point.
```

Output appears:

This sample is set in English Times 8-point. There are 30 sizes allowed, from 5 point to 72 point. The width of the column measure to be used often determines the most appropriate size of type to use. This manual is set primarily in 11 point.

Example 4(b)

Source file contains:

```
[*cg8400]
[c16.0]
[p12]
This sample is set in English Times 12-point. There are 30 sizes allowed, from 5
point to 72 point. The width of the column measure to be used often determines the
most appropriate size of type to use. This manual is set primarily in 11 point.
```

Output appears:

This sample is set in English Times 12-point. There are 30 sizes allowed, from 5 point to 72 point. The width of the column measure to be used often determines the most appropriate size of type to use. This manual is set primarily in 11 point.

Again, different sizes can be used in the one document:

Example 4(c)

Source file contains:

```
[*cg8400]
[c24.0][p12]
12 [p11]11 [p10]10 [p9]9 [p8]8 [p7]7 [p6]6 [p5]5 ... [p20]POW!
```

Output appears:

12 11 10 9 8 7 6 5 ... POW!

EXAMPLE 5 — CHANGING VERTICAL SPACING (LEADING)

Introduction: In the last set of examples, a variety of pointsizes were used but the distance between successive pairs of lines was kept constant, so that the output from Example 4(b) appears more "cramped" that that from Example 4(a), because of the larger pointsize used in that case.

For best visual results, pointsize and "space between lines", called *leading* (and pronounced "ledding"), should be considered together.

Leading is measured *strictly* (unlike pointsize) in points (or points and tenths of points).

The command used has the form **[v*?*]**, where *?* denotes the required value of the leading.
(**v** indicates "vertical" space, or leading)

Thus, [v9] denotes 9 point leading, and
[v10.5] denotes 10½ point leading (i.e. 10.5 points from baseline to baseline of successive lines).

If leading is not specified by the user, the system will assume the value 12, i.e. 12 point leading is the system default.

Example 5(a)

Source file contains:

```
[*cg8400]
[c12.0][p10][v13]
These samples show the visual effect (including depth)
of altering leading. In each case, the text used and the column
measure used have been kept the same, so that comparisons can be made.
```

Output appears:

These samples show the visual effect (including depth) of altering leading. In each case, the text used and the column measure used have been kept the same, so that comparisons can be made.

Example 5(b)

Source file contains:

```
[*cg8400]
[c12.0][p10][v11]
These samples show the visual effect (including depth)
...
```

Output appears:

These samples show the visual effect (including depth) of altering leading. In each case, the text used and the column measure used have been kept the same, so that comparisons can be made.

Example 5(c)

Source file contains:

```
[*cg8400]
[c12.0][p10][v10]
These samples show the visual effect (including depth)
...
```

Output appears:

These samples show the visual effect (including depth) of altering leading. In each case, the text used and the column measure used have been kept the same, so that comparisons can be made.

Note: Except to obtain special effects, the leading value should not normally be set to less than the pointsize value. The actual value chosen should depend not only on pointsize but also on typeface and width of page.

INTERLUDE

The five parameters so far discussed, viz.

[*cg8400] *or* **[*aps]**	Defines output device
[c.?]	Defines measure in picas.points
[p.?]	Defines pointsize
[f.?]	Defines font (i.e. typestyle)
[v.?]	Defines leading (i.e. inter-line spacing) in points

should be considered the "basic" parameters to be specified *prior* to any text in a source file destined for typesetting.

The decisions regarding which typeface to use for any particular job, which pointsize, what measure, etc. do not have to be made before text can be entered into the source file; but they *must* be made *before* the file can be *processed* by the JUSTIF program.

Such decisions depend on visualizing in advance how the final product will appear—this is subjective, of course, and different users will often have widely-differing opinions on the relative merits of different settings of the same document.

For general document work, until you have more experience with the different typefaces, etc., it is probably safer to suggest conservatism rather than adventure with regard to choice of style!

The remaining examples demonstrate a range of other features which are commonly employed in setting a variety of document types, and how these may be obtained using this system. A more complete exposition (with associated explanation) of each of the available commands is given in Part B of this manual.

EXAMPLES 6 — PARAGRAPHS

Introduction: Normal documents do not consist of just a single paragraph, as our examples so far have done. ITPS allows us to define the basic parameters to be applied to all subsequent paragraphs (i.e. how much to indent the first line of new paragraphs, what vertical space (if any) to leave above new paragraphs, etc), and then to indicate in the file the conclusion of each paragraph.

The "parameter" command has the form **[ip***?1,?2,?3***]**, where

?1 denotes the size of indent (in picas.points notation) to be used on the first line of each new paragraph,

?2 denotes the amount of leading (*in addition to the normal leading*), in points, to be used immediately before the first line of any new paragraph,

?3 denotes the number of points which must still remain on the current page if a new paragraph is to be begun on that page. (See Examples 12 later for pagination.)

Note that this command simply *sets the values of* the parameters ... it doesn't cause anything to happen.

The "execution" command, which (i) marks the end of paragraphs, and then (ii) applies the values of the "parameter" command to the next line, has the form **/p**.

(In both commands, **p** denotes "paragraph")

Example 6(a)

Source file contains:

```
[*cg8400]
[f110][p9][v10][c14.0]
[ip0.9,0,0]
This text represents more normal
input for general setting, which
breaks the lengthy text into units
called paragraphs. With this system,
we are able to define the basic
style of all the paragraphs by means
of one command, and mark paragraph
endings by means of a second
command./p
We now begin a
new paragraph, which again is
terminated by use of the //p command./p
This ends the current demonstration./p
```

This command indicates that new paragraphs have an initial line indented by 9 points, with no added space between paragraphs, and no special testing for when new paragraphs are allowable.

The command /p indicates the end of a paragraph.

Output appears:

This text represents more normal input for general setting, which breaks the lengthy text into units called paragraphs. With this system, we are able to define the basic style of all the paragraphs by means of one command, and mark paragraph endings by means of a second command.

We now begin a new paragraph, which again is terminated by use of the /p command.

This ends the current demonstration.

Example 6(b)

Source file contains:

```
[*cg8400]
[f110][p9][v10][c14.0]
[ip1.6,5,0]
This text represents more normal
input for general setting, which
breaks the lengthy text into units
called paragraphs. With this system,
we are able to define the basic
style of all the paragraphs by means
of one command, and mark paragraph
endings by means of a second
command./p
We now begin a
new paragraph, which again is
terminated by use of the //p command./p
This ends the current demonstration./p
```

New-paragraph indent 1 pica 6 points, inter-paragraph spacing 5 points, and begin new paragraph anytime.

Output appears:

This text represents more normal input for general setting, which breaks the lengthy text into units called paragraphs. With this system, we are able to define the basic style of all the paragraphs by means of one command, and mark paragraph endings by means of a second command.

We now begin a new paragraph, which again is terminated by use of the /p command.

This ends the current demonstration.

EXAMPLES 7 — SIMPLE INDENTING

Introduction: A common strategy to emphasize certain sections of material is to indent these sections, i.e. to move them relative to either or both of the normal margins between which the type is set. ITPS commands exist to allow us to indent text on the left side only (a left-indent), or the right-side only (a right-indent), or both sides simultaneously (a both-sides-indent).

The command formats for these features are, respectively, **[il*?*]**, **[ir*?*]** and **[ib*?*]**, where (in each case) *?* indicates the magnitude of the indent in the picas.points notation.

Additionally, there are commands to allow us to specify the position in text at which any existing indent is to be cancelled. These are **[il%]**, **[ir%]** and **[ib%]** for the three indent-types.

Note: The symbol % within a command in ITPS is always used to indicate cancellation of that command-type.

Source file contains:

```
[*cg8400]
[f100][p9][v10][c20.0]
[ip0,0,0]
This paragraph is being set to the full
measure specified, so that
we can observe where the regular
margins are situated, and hence make any desired comparisons and
measurements./p
[il5.0]This paragraph will be
indented by 5 picas on the left side
because of the command preceding it.
Notice that we still have alignment, but the
point of alignment is changed, until the
indent is cancelled./p
[il%][ir3.0]Now the left indent has been
cancelled, but a right indent of 3 picas
has been substituted instead, as we can check from the output./p
[ib5.6]Finally, a both-sides indent has
been called for. Notice that the original right-indent value
has been replaced by the value requested in the new
indent command, viz. 5.5 picas./p
[ib%]At last we are back to where we started,
showing that elementary indents are not too difficult to achieve./p
```

Output appears:

This paragraph is being set to the full measure specified, so that we can observe where the regular margins are situated, and hence make comparisons and measurements.

> This paragraph will be indented by 5 picas on the left-side because of the command preceding it. Notice that we still have alignment, but the point of alignment is changed, until the indent is cancelled.

Now the left indent has been cancelled, but a right indent of 3 picas has been substituted instead, as we can check from the output.

> Finally, a both-sides indent has been called for. Notice that the original right-indent value has been replaced by the value requested in the new indent command, viz. 5.5 picas.

At last we are back to where we started, showing that elementary indents are not too difficult to achieve.

EXAMPLE 8 – QUADDING COMMANDS

Introduction: In Chapter A1, the idea of a "unit" of text was introduced—a "unit" being a paragraph, or a heading, etc. The final line of any "unit" is always treated somewhat differently to other lines of the "unit". All other lines are generally "stretched" to fit exactly between the margins defined by the **[c***?***]** command—this process is called *justification*. However, the final line of a "unit" is normally *not* justified, i.e. the inter-word spacing is not modified; usually, this "part-line" is pushed across to the left margin (as shown by the final line of each paragraph in Examples 6 and 7)—the official term is that it is set *quad left* or *flush left*. But sometimes, especially where the "unit" is a heading, we may wish to position it differently—perhaps we would like to push this piece of text to the right (quad right or flush right), or to position it in the centre of the line (quad centre).

There are commands for all three operations, viz.:

/l to position flush left
/c to centre
/r to position flush right.

Each command is placed immediately *following* the text to which it is to be applied.

Note: **/p** performs the same function as **/l** as far as the text preceding the command is concerned. But, it also acts on the following line, whereas **/l** does not (except for causing it to be placed on the next output line).

Source file contains:

```
[*cg8400]
[f100][p10][v11][c16.0]
This represents a text file
containing not only normal paragraphs but
also styled headings, and other displayed features. These displays
have been achieved with a mixture of different
type styles and different quadding commands./l
[f102]Sample Heading[f100]/c
The paragraphs are terminated with the ''quad left'' code, whilst
the heading is followed with the
''quad centre'' code. The final lines show the use
of the ''quad right'' code./l
[f101]Signature/r
Title/r
```

Output appears:

This represents a text file containing not only normal paragraphs but also styled headings, and other displayed features. These displays have been achieved with a mixture of different type styles and different quadding commands.

Sample Heading

The paragraphs are terminated with the ''quad left'' code, whilst the heading is followed with the ''quad centre'' code. The nal lines show the use of the ''quad right'' code.

Signature
Title

EXAMPLES 9 – INSERTING BLANK SPACE

Introduction: To produce a visually-pleasing document, it frequently happens that it is necessary to introduce "blank space" (or "whitespace") to avoid a "crammed" appearance.
Referring to Example 8 above, the general effect would have been improved if the heading had been made more dominant, and two possible ways (apart from the typeface change already shown) of achieving this include (i) increasing the pointsize for the heading, and (ii) adding "whitespace" around the heading. To perform option (ii), we could just change the value of the leading for the single heading line and then restore it when the next paragraph resumes.
However, there exists a special command which allows us to advance the film in the phototypesetter immediately (but once only) by any specified amount in points (and tenths of points).

This command has the form **[va*?*]**, where *?* denotes the amount to advance.
(**va** indicates "vertical leading advance")

Example 9(a)
(Modifying the contents of the file used in Example 8:)

Source file modified to read:

```
...
[va11][f102]Sample Heading[f100]/c
[va5.5]The paragraphs are terminated with the ''quad left'' code, whilst
...
```

Output appears:

This represents a text file containing not only normal paragraphs but also styled headings, and other displayed features. These displays have been achieved with a mixture of different type styles and different quadding commands.

Sample Heading

The paragraphs are terminated with the ''quad left'' code, whilst the heading is followed with the ''quad centre'' code. The nal lines show the use of the ''quad right'' code.

Signature
Title

As shown in the following example, this command takes effect *immediately*—even in the middle of a line of text (or the middle of a word!).

Example 9(b)

Source file contains:

```
[*cg8400]
[f100][p10][v11][c16.0]
This is a normal line of text./l
This line [va3]F[va3]A[va3]L[va3]L[va3]S [va3]away./l
Normal text again./l
```

Output appears:

This is a normal line of text.
This line $F_{A_{L_{L_S}}}$ away.
Normal text again.

We may also "advance" the film by a *negative* amount, by a "reverse leading" command.

This command has the form **[vr*?*]**, where *?* is the desired amount of vertical movement (in points and tenths of points) in the reverse direction.

(**vr** denotes "vertical leading reverse")

Example 9(c)

Source file contains:

```
[*cg8400]
[f100][p10][v12][c12.0]
This text is set 10 point on 12 point leading, i.e. there
should be 12 points between successive baselines./l
[p8][v9][vr3]The leading has been changed to
9 point to match the change to 8 point type.
The first line of this new style has been reversed
by 3 points too./l
```

Output appears:

This text is set 10 point on 12 point leading, i.e. there should be 12 points between successive baselines.

The leading has been changed to 9 point to match the change to 8 point type. The first line of this new style has been reversed by 3 points too.

The final illustration combines both the **[va*?*] and [vr*?*]** commands with pointsize changes to show how inferior and superior characters (i.e. subscripts and superscripts) could be generated when these are not directly available as characters in the chosen font.

Example 9(d)

Source file contains:

```
[*cg8400]
[f100][p10][v11][c20.0]
This file demonstrates the creation of
inferior characters, such as might be used in
a chemical formula, e.g.
H[va2][p7]2[vr2][p10]SO[va2][p7]4[vr2][p10], and
superior characters, used to indicate footnotes[vr3][p7]23[va3][p10] within
the body of text./l
```

Output appears:

This file demonstrates the creation of inferior characters, such as might be used in a chemical formula, e.g. H_2SO_4, and superior characters, used to indicate footnotes^{23} within the body of text.

EXAMPLE 10 — SIMPLE TABLES

Introduction: ITPS allows us to define (in a variety of ways, one of the simplest of which is discussed below) tab stops, so that tabular material (i.e. tables) can be displayed. Once tab stops have been so defined, different special quadding commands are available which position the preceding text within the current "sub-column" (i.e. between tab stops) and begin entering the subsequent text in the next available "sub-column".

Note: Tab stops defined in this way do *not* allow continuous text to be displayed in multi-column form; other commands and techniques using indent commands are required for this.

The commands relevant to tab setting include:

[ts*?1,?2,?3,* ... **]**, which defines the positions of the tab stops.
?1, ?2, ?3, ... are measurements in the picas.points notation from the left margin.

[t%], which cancels all previously-defined tab stops.
Note: This is very important where one set of tab stops is to be replaced by another, because successive tab stop commands are *cumulative*.

/u, **/v**, **/w** are the common quadding commands, used in tabular displays, which correspond to the **/l**, **/c**, **/r** commands previously introduced.

Source file contains:

```
[*cg8400]
[f100][p10][v11][c24.0]
[t%][ts5.0,12.6,18.0]
Item 1/uPointsize/uExample 4/uEasy/l
Item 2/uIndents/uExample 7/uEasy/l
Item 3/uMultiple columns/uExample 12/uHarder/l
```

Output appears:

Item 1	Pointsize	Example 4	Easy
Item 2	Indents	Example 7	Easy
Item 3	Multiple columns	Example 12	Harder

EXAMPLE 11 — LEADERS

Introduction: In the setting of material such as a contents page of a document or a pro-forma, it is common to find a style in which some text is set flush left on a line, other text is set flush right on the same line, and the intervening space (which may vary considerably from line to line) is filled with some character, which may assist the eye to identify correctly the corresponding left- and right-side elements, e.g.

Chapter Title ..... Page Number
Next Title 29

This style is referred to as the insertion of *leaders*, and can be easily introduced using the ITPS system.

The commands needed to produce such output are:

/. which passes the instruction to use a system-defined leader-character instead of normal whitespace *for the current line only*.

/q, another special quadding command, which forces preceding text to be set flush left, but allows other text to be set *on the same line* with its own separate quadding command.

Example 11

Source file contains:

```
[*cg8400]
[f100][p10][v11][c12.0]
Left entry/q/.Right entry/r
Next/q/.More on right/r
```

Output appears:

Left entry.....Right entry
Next ..... More on right

EXAMPLE 12 — MULTIPLE COLUMNS

Introduction: The facility exists to allow us not only to reverse a nominated number of points (using the **[vr.?]** command) but also to reverse to some previously specified position. This facility is useful especially in multiple-column displays, and can be manipulated so that a form of automatic pagination can be achieved.

The commands involved are:

[vg.?.?], where *?.?* denotes the size of the "gutter"i.e. inter-column space, (in picas.points notation) to be inserted between columns.

[vc?], where *?* denotes the *depth* (in points) of each column before beginning a fresh column. (This is a simplified form of the full command, which is given in detail in Part B of this manual.)

[cm.?.?], where *?.?* denotes the total width in picas.points notation to be made available to the complete multiple-column display. (The **[cm?.?]** command gives the *maximum* measure for setting, which must account for all the columns (each of which takes the current "column measure" (i.e. the value in the normal **[c.?]** command) as the width of a single column) *plus* all the necessary "gutters" between text columns. The system then determines how many columns can fit in the maximum measure, and so needs to know only how deep each column must be.)

[vc%], which cancels any current **[vc.?]** command.

Example 12(a)

Source file contains:

```
[*cg8400]
[f100][p10][v11][cm24.0]
[vg2.0][c10.0]

[vc99]
This sample will demonstrate how to set
simple multiple-column continuous text.
Since we have defined the leading to be 11 point, and the column
depth to be 99 points, we are really requesting that a new column
begin after 9 lines have been set.
Since two 10-pica columns with a 2-pica separating
gutter can fit into a maximum column width set to 24 picas, we
expect the output to be set ''two-up'' across the page.
Notice that at the end of the ''right-hand'' column, text reverts
back to the left-column again (with a small clearance gap)
and the pattern repeats.
This process continues indefinitely, and
forms the basis for pagination, shown in the
next example.
```

Defines *total* width

Defines gutter of 2 picas and each column to be 10 picas wide. So 2 columns will fit into 24 picas.

Defines depth of columns.

Output appears:

This sample will demonstrate how to set simple multiple-column continuous text. Since we have defined the leading to be 11 point, and the column depth to be 99 points, we are really requesting that a new column begin after 9 lines have been set. Since two 10-pica columns with a 2-pica separating gutter can fit into a maximum column width set to 24 picas, we expect the output to be set ''two-up'' across the page. Notice that at the end of the ''right-hand'' column, text reverts back to the left-column again (with a small clearance gap) and the pattern repeats. This process continues indefinitely, and forms the basis for pagination, shown in the next example.

As a special case of the above technique, if the value in the **[cm.?]** command is set so that only one column (as specified in the **[c.?]** command) can fit, then at the end of the first single column, a clearance gap will be inserted, and the second column will begin immediately after the first, i.e. we have pagination (of sorts, because we haven't considered "running heads" or "folios").

Example 12(b)

Source file contains:

```
[*cg8400]
[f100][p10][v11][cm15.0][vg2.0][c12.0][vc88]
This time, with maximum-column set to 15 picas, normal
column measure set to 12 picas, and gutter-width set to 2 picas,
only a single column can fit. So each column will
set below its predecessor, as single pages.
Because depth has been set to 88 points, and 11-point
leading is being used, each ''page'' will consist of
8 lines of text.
```

Output appears:

This time, with maximum-column set to 15 picas, normal column measure set to 12 picas, and gutter-width set to 2 picas, only a single column can fit. So each column will set below its predecessor, as single pages. Because depth has been set to 88 points, and 11-point leading is being used, each ''page'' will consist of 8 lines of text.

A more difficult example

Where columns of *unequal measure* are required, we need to use more complex techniques, involving the indent commands (see Examples 7) together with the commands given below:

These additional commands have the forms:

[vm.?], where *?* is a value from 0 to 9.
(**vm** denotes "mark vertical position".)

[vp.?], where again *?* is a value in the range 0-9.
(**vp** indicates to "vertically position" at the marked position.)

These two commands are used in conjunction in the following way:

Suppose we may need to return to some definite position in text at a later time. Then, we "mark" that position with a **[vm.?]** command (i.e. there are 10 different "spots" we can mark). When we need to return to that position, we call up the **[vp.?]** command using the same value. This technique is especially useful when it is difficult to keep track of how far we could have moved from the marked spot in the interim—remembering we are always working "blind".

Example 12(c)

Source file contains:

```
[*cg8400]
[f100][p10][v11][c12.0]
This is an initial paragraph in the usual
form. We will "mark" the beginning of
the next paragraph./l
[vm1]This can be referred to as
"mark point 1", and we can go back to that spot
at any later time./l
[vp1][c21.0][il14.0]The sequence of commands
inserted at the start of this paragraph will
(a) send us back to the first marked position,
(b) re-set the column measure to 21 picas,
(c) indent 14 picas on the left side.
The net effect of all this is to give us a
7-pica column beside a 12-pica column./l
```

Output appears:

This is an initial paragraph in the usual form. We will "mark" the beginning of the next paragraph.

This can be referred to as "mark point 1", and we can go back to that spot at any later time.

The sequence of commands inserted at the start of this paragraph will (a) send us back to the first marked position, (b) re-set the column measure to 21 picas, (c) indent 14 picas on the left side. The net effect of all this is to give us a 7-pica column beside a 12-pica column.

EXAMPLE 13 — SIMPLE MACROS (or HOW TO MAKE YOUR OWN COMMANDS)

The list of commands necessary to achieve a particular effect may be very lengthy and, if this effect occurs frequently in a document, the constant insertion of a long string of commands is both tedious and likely to lead to occasional inconsistencies.
For example, a particular style of common sub-heading in an article may require a sequence of commands such as

[il3.0][va24][f112][p14][v14]

to precede the text of the heading, and an equally difficult sequence to conclude the heading, such as

/l[il%][va12][f110][p11][v12]

To avoid these problems, we may choose to create our own "home-grown" commands, so that a single command can replace a lengthy and/or difficult sequence.

Note: This is worthwhile only when the command-string is to be used frequently.

To generate "simple" commands to correspond to the sequences of commands illustrated above, we create (with an editor) two files, both of which must have the extension .MCR. Since our two sequences are concerned with the commands to *begin a heading* and to *end a heading*, we shall choose to call the two files BH.MCR and EH.MCR respectively.

Then, each file simply contains the same sequence of commands which we would have used to perform the job directly in the source file, i.e.

BH.MCR consists of the single line
[il3.0][va24][f112][p14][v14]
and EH.MCR consists of the single line
/l[il%][va12][f110][p11][v12]

To *use* these simple commands (or *macros*), we include in the source file (at the same points we would have issued the full string of standard commands) the commands
[*bh] and **[*eh]** respectively.

Note: The form of these commands includes an asterisk, which precedes the name of the command.

For users preparing a class of documents, all of which have the same basic format, the use of personal commands can reduce both the work involved and the risk of inconsistency.

Example 13

(This example uses the macros defined above.)

Source file contains:

```
[f100][p11][v12]
...
end of section./l
[*bh]New Section Heading[*eh]
This is start of new section ...
```

Output appears:

...
end of section.

New Section Heading

This is start of new section ...

PART B

DETAILED DESCRIPTION OF COMPOSITION COMMANDS

CHAPTER B1
Introduction to the ITPS Composition System

The ITPS commands are inserted into the body of a text file by the user to control the appearance of the typeset product.

In the ITPS system, all commands take one of the following forms:

[command-code] *or*
[command-code *value(s)*] *or*
/command-code

In the following chapters of Part B, each available command will be considered separately, and often in combination with other commands, to illustrate common uses of the command and restrictions which apply to the command. In all cases, examples are provided as models which newer users may follow in designing their own documents.

ACCESS CODES

As well as commands, which affect the positioning of characters, ITPS also demands that "non-standard" characters be called up by codes, which in this manual will be referred to as *access codes*.

All "access codes" have one of the forms:

+ x or
/x,

where, in either case, *x* is a "standard" (i.e. alphabetic, numeric or punctuation) character.

The access codes may vary from font to font in meaning, e.g. in English Times face the code **+ k** will result in the production of the character ©, whereas in Serif Gothic style, the same access code **+ k** will produce the character ķ. The user should refer to Appendix C and Appendix E for a complete list of all available characters in all available fonts, together with the appropriate access code for each.

FORMAT OF PART B

In the remainder of this Part, the following conventions will be adopted:

1. Type set in Univers style will represent exactly what the user would type in his/her file ... text and commands.
   This is a sample of Univers style.
2. The resultant output will *generally* be displayed in the style called English Times. (It will not always be possible to adhere to this standard ... in particular, when the output is supposed to have been generated with some other style in accordance with the ITPS command in the input file.)
   This is a sample of English Times.

CHAPTER B2
Basic Set-Up Commands

In order to produce phototypeset material, certain fundamental decisions must be made concerning the basic style desired, and the commands necessary to effect this style must then be incorporated into the input (source) file as a command sequence preceding the first text characters in the file. These decisions are:

- the *output device* to be used—generally the COMPUGRAPHIC 8400 machine, but possibly the APS-5 machine located at Queensland Newspapers or (less frequently) a pseudo-typesetting device such as a Diablo terminal or line-printer
- the initial *typeface* to be used
- the *pointsize* to be employed initially
- the *leading* (i.e. vertical space between lines)
- the *measure* (or width) to which the initial copy is to be set.

Note: If any of the terms above are not familiar to you, it is suggested that you read Part A of this manual.

Remember that, unless default values are being used, the five commands corresponding to the five items above *must* be inserted at the very beginning of the file to be processed.

Introductory Example

Source file contains:

	Comments
[*cg8400]	Output to COMPUGRAPHIC 8400
[f100]	Use English Times face
[p9]	Use 9-point type,
[v10]	on 10-point leading
[c16.0]	Set 16 picas wide
This short demonstration file illustrates the basic commands to be inserted at the beginning of any file destined for typesetting. If any are omitted (except the output device command), a system default value will be used. Text is input freely, with no thought of nal appearance ... this depends on the commands inserted, not on the "appearance" of the source file./l	The command /l will be explained later.

Output appears:

This short demonstration file illustrates the basic commands to be inserted at the beginning of any file destined for typesetting. If any are omitted (except the output device command), a system default value will be used. Text is input freely, with no thought of final appearance ... this depends on the commands inserted, not on the "appearance" of the source file.

Each of the above commands will be discussed in more detail in subsequent chapters of this manual.

CHAPTER B3
Output-Device Commands

Since output can be directed to any one of a variety of devices, each with its own distinctive set of characteristics (available typefaces, available characters, etc), it is necessary to advise the program JUSTIF of the intended destination of the file.

The available devices, with the corresponding commands, are listed below:

1. [*cg8400] COMPUGRAPHIC 8400 phototypesetter (at Prentice Computer Centre)
2. [*aps] APS-5 phototypesetter (at Queensland Newspapers)
3. [*diablo] DIABLO terminal
4. [*lpt] Normal lineprinter

Notes:

1. Only the COMPUGRAPHIC 8400 and APS-5 are genuine phototypesetting machines. The others do not have the capabilities normally expected in a typeset product, e.g. typeface changes, different sizes of type, varying character widths, etc.
2. A number of typefaces are common to both phototypesetting machines, but each does have its own unique faces as well. Also, the characters available on the two machines are different. Simply changing the output-device command in the source file may result in unexpected changes in the output, because of the different character-sets.
3. Unless there is a special reason for doing so, all files intended for typesetting should be directed to the COMPUGRAPHIC 8400 machine.

It is *very important* to be aware of the fact that there is *no default* for this command *and* it must precede all text and all commands in the file. If the program JUSTIF is passed a file in which the first item in the file is not one of the above commands, then that file will be rejected.

CHAPTER B4
Typeface Commands

The available characters in any style (character-shape) are organized into *fonts*, i.e. collections of characters of a common style. The available "text" styles include English Times (in a variety of text, italic, bold, bold italic, condensed and display forms), Baskerville, Mallard, Univers, Helios, Souvenir, Tiffany, University, Serif Gothic, Korinna Extrabold and Goudy Bold; in addition, there are "specialist" styles involving Greek-Mathematics symbols, phonetic symbols, and "logos".

The complete list of fonts is shown in Appendix A, with a display for comparisons of style in Appendix B, and a character-listing in both Appendix C and Appendix E.

Each font is identified by a unique "font number", as given in Appendix A.

[f?] COMMAND

Any available typeface can be called up at any time by the command

[f*?*], where *?* is the font number of the desired typeface.

The typeface so called up remains in effect until superseded by a later **[f*?*]** command, or special associated typeface command given later in this chapter.

Notes:

1. If no **[f*?*]** command is included near the beginning of the file (see Chapter B2), then the parameter *?* takes the default value 100. Referring to the table of Appendix A, this means that the default typeface is English Times.
2. If an illegal value of *?* is used (i.e. a value not given in Appendix A), then an error message is displayed during processing by JUSTIF, and no typeface change is effected.

Example

Source file contains:

```
...
[f100]
The initial text is English Times Medium, but this
may be changed to [f102]bold style[f100] or
[f101]italic style[f100] at any time.
```

Output appears:

The initial text is English Times Medium, but this may be changed to **bold style** or *italic style* at any time.

Point of Style

Except in special display work (e.g. advertising), it is generally accepted that all material in the one article, or book, or journal, etc. should be set essentially using only faces of the one "family", i.e. if the basic style is, say, English Times, then use only English Times, Times Bold, Times Italic, Times Bold Italic, etc ... do not mix in styles from other families such as Baskerville or Souvenir or Helios, etc.

Notwithstanding the above remark, there will be occasions when a break to some different family (e.g. Univers) may be considered visually desirable for, say, headings. This is essentially a question of aesthetics and therefore subjective—but, if in doubt, it is probably better to err on the side of conservatism.

Auxiliary Typeface Change Commands

While any typeface change can be effected by means of just the **[f?]** command, other typeface commands are available for convenience in particular situations, and these are discussed below.

[fb] AND [fi] AND [fs] COMMANDS

These commands call up the bold, italic and bold italic faces respectively corresponding to the typeface currently in effect (provided this related member of the typeface family exists).

Thus,

both [f100]Here is [f102]bold text
and [f100]Here is [fb]bold text

produce Here is **bold text**

Similarly,

both [f100]Here is [f101]italic text
and [f100]Here is [fi]italic text

produce Here is *italic text*

Likewise,

both [f100]Here is [f103]bold italic text
and [f100]Here is [fs]bold italic text

produce Here is ***bold italic text***

Notes

1. If the face called by any of **[fb]** or **[fi]** or **[fs]** does not exist (e.g. if **[f160]**—Rockwell Medium—is being used, there is no corresponding italic face (See Appendix A)), then, during the processing by JUSTIF, an error message will be displayed, and the typeface change command will be ignored.
2. For all families of faces, the "medium" face is treated as the "regular" face. Where a "light" version of the face exists (e.g. Univers), this "light" face can only be called up by means of the **[f?]** command.
3. The face invoked by any of the commands **[fb]** or **[fi]** or **[fs]** remains in effect until replaced by a face called up by either of the two commands **[f?]** or **[fr]** (see below).
4. Note especially that the "regular" face should be the current face when the commands **[fi]**, **[fb]** or **[fs]** are used.

Another command of this type, viz. **[fv]**, also exists for *reversing* the current face. This command is discussed and demonstrated in Chapter 17.

[fr] COMMAND

Whenever **[fb]** or **[fi]** or **[fs]** is used, not only does the current typeface change, but also the previous font number is stored by the system. This original typeface may be *restored* as the current typeface by the command **[fr]**.

Thus,

both [f100]Here is [fb]bold text[fr] followed by normal style

and [f100]Here is [f102]bold text[f100] followed by normal style

produce Here is **bold text** followed by normal style

ONE-LINE TYPEFACE CHANGES — [wf?], [wb], [wi], [ws]

Again, for convenience, where a typeface change is required for just a single output line (e.g. a heading), instead of using

either
```
[f---]... normal text
[f---]Heading[f---]
Normal text again
```
or
```
[f---]... normal text
[fb]Heading[fr]
Normal text again
```

we may use any of the commands **[wf?]** or **[wb]** or **[wi]** or **[ws]** in the following way:

```
[f---]... normal text
[wb]Heading
Normal text again
```

i.e. the typeface change called up by any of these commands remains in effect for just a single output line, and then the typeface reverts automatically to the original face.

[fm] AND [fm%] COMMANDS

Certain phototypesetting machines, including the COMPUGRAPHIC 8400 and APS-5, have the capability to "slant" characters.

This is achieved by the insertion of the command **[fm]** *immediately before* the first character to be slanted, and cancelled by inserting the command **[fm%]** immediately following the last character to be slanted,
i.e. all characters between the commands **[fm]** and **[fm%]** will be slanted.

Example

Source file contains:

```
[f500]Normal text [fm]followed by slanted,[fm%] followed by normal.
```

Output appears:

Normal text *followed by slanted,* followed by normal.

Warning:
While this command may be used with any typeface without error, it is *not* the general method for achieving italic text, except with special fonts (viz. the sans-serif fonts) such as Helios. With other families, the italic face has many differences from the "regular" face apart from its "slant".

CHAPTER B5
Pointsize Commands

The characters available in any font may be displayed in any one of 30 available sizes of type, ranging from 5-point to 72-point.

The available sizes are:

5	6	7	8	8.5	9	9.5	10	10.5	11
11.5	12	13	14	15	16	18	20	22	24
26	28	30	36	42	48	54	60	66	72

Note: Pointsize is measured in units of the point. However, this measure is not a *strict* reference to the printing unit called a point (where 72 points are equivalent to one inch). Rather, 8-point type, say, is (arbitrarily) type of such vertical dimension that it will "comfortably" sit between baselines separated by 8 points of space. But, character height of the same pointsize will vary considerably with typeface.

[p.?] COMMAND

Any legitimate pointsize is called up at any time by the command

[p.?], where .? denotes any value from the list above.

The pointsize so called remains in effect till replaced via a subsequent **[p.?]** command (or a **[wp.?]** command—see later in this chapter).

Note: If the **[p.?]** command is omitted from the beginning of a file then the initial text, by default, is set in 10-point.

Example 1

Source file contains:

```
[p9]This is an example showing [p12]LARGER[p9] and
[p6]smaller[p9] type.
```

Output appears:

This is an example showing
LARGER and smaller type.

Example 2

Source file contains:

```
[p9]T[p10]H[p11]I[p12]N[p13]K [p14]B[p15]I[p16]G
```

Output appears:

THINK BIG

Example 3

Source file contains:

```
[p18]T[p7]HIS STYLE[p10] demonstrates the use of
a larger character as the initial character of a chapter, and also the
style called ''small caps'' (and how to manufacture these when they are
not directly available in a font).
```

Output appears:

THIS STYLE demonstrates the use of a larger character as the initial character of a chapter, and also the style called "small caps" (and how to manufacture these when they are not directly available in a font).

Note: The reader should also refer to the modified form of the pointsize command discussed in Chapter B6.

Auxiliary Pointsize Commands

ONE-LINE POINTSIZE COMMAND — [wp.?] COMMAND

This command changes pointsize from its current value to the value nominated in the **[wp.?]** command, but *only for the remainder of the current line*, after which the original pointsize is automatically restored.

(a) Issued at beginning of line

Source file contains:

```
[p9]... end of paragraph./I
[wp12]HEADING/I
New paragraph ...
```

Output appears:

... end of paragraph.

HEADING

New paragraph ...

(b) Issued in middle of line

Source file contains:

```
[p9]This paragraph contains a one-line-only
pointsize command [wp12]in the middle of
a line, so that it will remain in effect only until
the end of the current output line.
```

Output appears:

This paragraph contains a one-line-only pointsize command in the middle of a line, so that it will remain in effect only until the end of the current output line.

Note: Again, as with the one-line typeface change commands, the **[wp.?]** command is useful in practice when setting single-line headings which require a pointsize change from the body text for emphasis.

Avoiding Illegal Pointsizes

Since there is a limit (30) on the number of available pointsizes, there is always the risk of the user attempting to select an illegal pointsize. In particular, for users designing their own typesetting macros (see Chapter B19), the very real possibility exists that a pointsize is chosen as the result of a calculation, which could lead to attempts to call a pointsize which is not available.

Three further pointsize commands are available to avoid this problem.

[pr.?], [pg.?] AND [pl.?] COMMANDS

(a) If the value *?* is legal (i.e. it is a member of the set of numbers at the beginning of this chapter), then each of these commands functions as the **[p.?]** command for the same value of *?*.

(b) Where the value *?* is not legal, each of these three commands substitutes pointsizes which *are* legal in the following ways:

(i) **[pr.?]** selects the *closest* legal pointsize to the nominated value,
e.g. **[pr7.9]** is replaced by **[p8]**
[pr7.1] is replaced by **[p7]**

(ii) **[pg.?]** selects the closest legal pointsize *greater than* the specified size,
e.g. **[pg40]** is replaced by **[p42]**
Note: If the nominated value is greater than 72 (the maximum legal size), then **[p72]** is applied.

(iii) **[pl.?]** selects the closest legal pointsize *less than* the specified value,
e.g. **[pl35]** is replaced by **[p30]**
Note: If the specified value is less than 5 (the minimum legal pointsize), then **[p5]** is applied.

CHAPTER B6
Leading Commands

Introductory Note: The term *leading* (pronounced "ledding") is defined as the "vertical" distance between successive lines of text, and is measured in points, or points and tenths of points. Commonly in books, journals, etc. leading is set 1 point greater than the current pointsize (but this can vary with certain typefaces, which may be "taller" than other typefaces of the "same" pointsize),
e.g. this manual is set primarily in 11-point type with 12-point leading. (This is frequently referred to as setting "11-point on 12-point" or just "11 on 12-point".)

The visual effect of using different leading is demonstrated below.

[v.?] COMMAND

Leading is set or changed by the command

> **[v.?]**, where *.?* is the desired leading value in points (e.g. **[v10]**), or points and tenths of points (e.g. **[v10.5]**).

If this command is issued at the beginning of a new output line, it takes effect immediately; if it is issued in the middle of an output line, it does not take effect until the beginning of the next output line. Any leading command remains in force until replaced by another command of the same type, but may *temporarily* be replaced (for one line of output) by the command **[wv]** (see later in this chapter) or suspended (for one occasion) by such commands as **/z** and **/q** (see in Chapter B8).

If the **[v.?]** command is omitted at the beginning of a source file, it is set by the system to the value 12.

Example

Below are shown a number of samples of the same text set in the same style, with the same pointsize, but with different values defined for the leading.

Source file contains:

```
[*cg8400]
[f100][p10]                    English Times 10-point
[v14]                          14-point leading
This is a series of demonstrations in
which the style and size of type are kept constant,
but the leading is progressively decreased, so that
the visual effect of leading changes can be observed.
```

Output appears:

This is a series of demonstrations in which the style and size of type are kept constant, but the leading is progressively decreased, so that the visual effect of leading changes can be observed.

Source file contains:

```
[*cg8400][f100][p10]          English Times 10-point
[v12]                         12-point leading
This is a series of demonstrations in
which the style and size of type are kept constant,
but the leading is progressively decreased, so that
the visual effect of leading changes can be observed.
```

Output appears:

This is a series of demonstrations in which the style and size of type are kept constant, but the leading is progressively decreased, so that the visual effect of leading changes can be observed.

Source file contains:

```
[*cg8400][f110][p10]          English Times 10-point
[v11]                         11-point leading
This is a series of demonstrations in
which the style and size of type are kept constant,
but the leading is progressively decreased, so that
the visual effect of leading changes can be observed.
```

Output appears:

This is a series of demonstrations in which the style and size of type are kept constant, but the leading is progressively decreased, so that the visual effect of leading changes can be observed.

Source file contains:

```
[*cg8400][f110][p10]          English Times 10-point
[v10]                         10-point leading ... called "solid" when pointsize and
This is a series of demonstrations in  leading equal
which the style and size of type are kept constant,
but the leading is progressively decreased, so that
the visual effect of leading changes can be observed.
```

Output appears:

This is a series of demonstrations in which the style and size of type are kept constant, but the leading is progressively decreased, so that the visual effect of leading changes can be observed.

Source file contains:

```
[*cg8400][f110][p10]          English Times 10-point
[v9]                          9-point leading
This is a series of demonstrations in
which the style and size of type are kept constant,
but the leading is progressively decreased, so that
the visual effect of leading changes can be observed.
```

Output appears:

This is a series of demonstrations in which the style and size of type are kept constant, but the leading is progressively decreased, so that the visual effect of leading changes can be observed.

Note: too cramped!!

MODIFIED POINTSIZE COMMAND — [p*?1*,*?2*]

Since pointsize and leading are closely related for best aesthetic effects, whenever pointsize is changed, consideration should always be given to the probable need to alter leading correspondingly. Both can be changed within the single modified pointsize command

[p*?1*,*?2***]**, where
?1 denotes the new pointsize to be used, and
?2 denotes the new leading to be used.

Example

Source file contains:

```
[p10,11]                                  Set 10-point type on 11-point leading
This paragraph is set 10-point on 11-point
leading. Both size and leading
will be changed for the next paragraph./l
[p12,14]This paragraph is set 12-pointSet 12-point type on 14-point leading
on 14-point leading, both parameters being
changed in the one command./l
```

Output appears:

This paragraph is set 10-point on 11-point leading. Both size and leading will be changed for the next paragraph.

This paragraph is set 12-point on 14-point leading, both parameters being changed in the one command.

IMMEDIATE LEADING COMMANDS — [va.*?*] AND [vr.*?*]

The two commands **[va.*?*]** and **[vr.*?*]** allow *immediate* movement, in either "vertical" direction, by the amount (number of points or points and tenths of points) specified in the command, without affecting the normal leading between lines.

To advance, use the **[va.*?*]** command:

(a) **Source file contains:**

```
This demonstrates [va5]the use of added leading
within a line of text
```

Output appears:

This demonstrates the use of added leading within a line of text

(b) **Source file contains:**

```
... end of section./l
[va10]New section begins with extra vertical
space preceding it for added emphasis.
```

Output appears:

... end of section.

New section begins with extra vertical space preceding it for added emphasis.

To reverse, use the **[vr.*?*]** command:

Source file contains:

```
This [vr3]shows [vr3]an [vr3]uphill [vr3]climb
```

Output appears:

This shows an uphill climb

The example below shows the use of both immediate leading commands (together with pointsize commands) to generate superior numbers (superscripts), where these do not occur as characters in the font in use.

Example

Source file contains:

```
[p10][v11]A superior number[vr3][p7]29[p10][va3] may be generated.
```

Output appears:

A superior number[29] may be generated.

Auxiliary Leading Commands

(i) ONE-LINE LEADING CHANGE — [wv.?]

This command causes leading to be changed for just a single line, after which it reverts to its previously-used value. This can be convenient where added space is needed above and/or below headings, between sections, etc.

Example

Source file contains:

```
[p10][v11][c12.0]
This paragraph is included to show normal setting
so that we may compare the changes caused
by the insertion of the new command.[wv16.5]/l
The new section is dropped 5.5 points
(i.e. half a line) but then the initial value
for the leading is used for all other lines.
```

Output appears:

This paragraph is included to show normal setting so that we may compare the changes caused by the insertion of the new command.

The new section is dropped 5.5 points (i.e. half a line) but then the initial value for the leading is used for all other lines.

CANCEL LEADING FOR ONE LINE — [vz] COMMAND

This command is a very special case of the **[wv.?]** command ... it sets the leading value to *zero* for the next line set.

Thus, it allows us to"print over" one line of text with other text.

Example

Source file contains:

```
[vz]Mistakes are easily made./l
[vz]xxxxxxxx/l
[vr9]Errors
```

Output appears:

Errors
Mistakes are easily made.

Note: Where "zero leading" is required, in practice it is generally simpler to use the command **/z**, described (with applications) in Chapter B8.

CHAPTER B7
Measure (Width) Commands

[c.?] COMMAND

This command defines the width of the line within which the ITPS-10 programs must perform the necessary hyphenation-justification functions. This width is referred to as the "column *measure*".

The parameter *?* is given in the picas.points notation, e.g.

[c12.0] requests that output be justified to a measure of 12 picas
[c18.6] requests justification to a measure of 18 picas and 6 points
[c22.10] requests justification to a measure of 22 picas and 10 points.

All subsequent text will be set to the nominated width, until amended by a later **[c.?]** command. The default value is 36 picas.

Example

Source file contains:

```
[c12.0]All text is initially set to a
measure of 12 picas, because of the inital width
command given./l
[c18.0]This paragraph is set half as wide again
because it is preceded by a column-width command
requesting an 18-pica measure./l
[c9.0]At any time, we can reduce the measure
also by asking for the appropriate width./l
```

Output appears:

All text is initially set to a measure of 12 picas, because of the initial width command given.
This paragraph is set half as wide again because it is preceded by a column-width command requesting an 18-pica measure.
At any time, we can reduce the measure also by asking for the appropriate width.

Note: In this chapter, the term "column" refers simply to "single-column" output. For multiple-column displays, the reader should refer to Chapters B12 and B18.

[cm.?] COMMAND

Because of the physical size of the phototypesetting machines, there is a maximum width of chemically-treated paper which can pass through these machines. This, in turn, restricts the width to which copy may be set. This maximum width is 70 picas.

However, the user may (and in some applications will have to) set the maximum width to a smaller figure than the default value of 70 picas.

This is done by the command

[cm.?], where *?* is the desired maximum measure. Like the parameter of the **[c.?]** command, *?* is given in picas.points notation.

The particular situation in which this command becomes necessary involves the setting of multiple columns of text ... see Chapter B18.

It should also be noted that, while there is no command for setting a minimum value for column measure, nevertheless the system *does* have such a restriction, viz. 2 picas, i.e. JUSTIF will not accept the task of attempting to justify copy in a column less than 2 picas wide.

[cz] COMMAND

This command may be used to cancel any indents or tabs (see Chapters B10–B12 for descriptions of these) and hence restore the full measure defined in the most recent **[c*?*]** command.

CHAPTER B8
Quadding Commands

If any collection of one or more whole or part lines is considered a "unit" (i.e. a "unit" may commonly be a paragraph or a heading, etc.), then we must issue a command to define how the final line of the "unit" is to be treated.

This is because the final line is special in that it is *not normally justified.*

THE COMMANDS /l AND /r AND /c

In the simplest terms, there are three ways of handling a final line:

(i) FLUSH LEFT
or QUAD LEFT

(ii) CENTRE
or QUAD CENTRE

(iii) FLUSH RIGHT
or QUAD RIGHT

The respective commands for these three treatments are:

/l to set the text flush left

/c to set the text centred

/r to set the text flush right

Example

Source file contains:

```
[c15.0]
Left Text/l
Centred Text/c
Right Text/r
```

Output appears:

```
Left Text
        Centred Text
                Right Text
```

Notes:

1. After any of the commands **/l** or **/c** or **/r**, the subsequent text will appear on a new output line.
2. Normally units of text such as paragraphs are terminated with either **/l** or **/p** (see later in this chapter). **/c** and **/r** are used chiefly for display purposes (e.g. headings in papers or names/addresses at beginnings or ends of documents).
3. "Unnecessary" quadding commands result in *blank lines* being output, e.g.

```
text A/l/l          results in          text A
text B/l
                                       text B
```

You may prefer to use the **[va.?]** command to control additional spacing, since it is more flexible in the amount of leading which may be added.

QUAD PARAGRAPH COMMAND /p AND [ip*?1,?2,?3*] COMMAND

One primary decision to be made in the styling of most documents concerns the basic appearance of paragraphs and their relation to surrounding text, viz.

- Is the introductory line of each paragraph to be indented? If so, by how much?
- Do we require extra leading between paragraphs? If so, how much?
- Are there are restrictions concerning when a new paragraph may be begun in relation to the bottom of a column or page?

Points of Style:
While there are no absolute guidelines for decisions such as these (since they involve personal preference and, even then, may vary from one document-type to another), nevertheless it should be stressed for the user not familiar with typesetting conventions that matter of a "continuous-text" nature tends to be set "tighter" than typed material, i.e.

- where initial lines of paragraphs are indented, the amount of such indent is normally small compared to the standards usually adopted in typing
- it is common (but not universal) practice not to include additional vertical space between paragraphs. (This decision is based largely on the type of document.)
- whereas in typing it is common to refrain from commencing a new paragraph if few lines of the new paragraph will fit on the current page, in typesetting it is standard practice in most documents to aim for pages of equal length, so that new paragraphs are normally begun even when only sufficient space exists on the current page to fit the opening line of the new paragraph. Where headings, for example, are involved, the decision will usually be quite different.

The command **[ip*?1,?2,?3*]** allows us to:

(i) pre-define these general characteristics of all paragraphs:
  (a) the parameter *?1* defines in the *picas.points* notation the amount by which the introductory line of each new paragraph is to be indented
  (b) the parameter *?2* defines *in points* the additional leading to be inserted between paragraphs
  (c) the third parameter *?3* defines least amount of vertical space (in *points*) which must remain in the current column or page (see Chapter B18) if the new paragraph is to be begun in that column or on that page. If less space remains, a new column or page is begun instead.

(ii) indicate the end of each paragraph by inserting the command **/p** at this point in text, which then causes the following two effects:
  (a) the preceding text on the current output line is set *flush left* on that line, and
  (b) the parameters defined in the preceding **[ip*?1,?2,?3*]** command are used to position the following text as a new paragraph relative to the paragraph just completed.

Note: **/p** is interpreted simply as **/l** unless a **[ip*?1,?2,?3*]** command is currently in effect.

Example

Source file contains:

```
[c12.0][ip0.10,6,0]
This is a short paragraph to
show how the ''quad para'' command
works./p
Note the extra leading forced
before the first line, and the
indent of the first line./l
```

Indent first line of any new paragraph by 10 points, and add 6 points extra leading between paragraphs.

Output appears:

This is a short paragraph to show how the "quad para" command works.

Note the extra leading forced before the first line, and the indent of the first line.

Note on Parameter ?3

Until you have need to use the commands discussed in Chapter B18, it is suggested that when using the command **[ip***?1*,*?2*,*?3***]** you should always set the parameter *?3* to the value zero.

Hanging Indents

Note: The general area of indenting is extended in Chapters B10–B11. Both here and there, users familiar with "word-processing" packages will have to learn to avoid "counting characters" when attempting indenting ... remember that, generally, any two given characters will have *different widths*.

Another common style for "units" of text is referred to as a "hanging indent", and this is illustrated below in a sample of typical bibliographic material.

Bell, M.W. (1979), *Typesetting For Beginners*, Creative Press, London.

Cross, P.A. and Double, W.P. (1978), *The Effects of Computerization on the Printing Industry*, John Brown and Sons, Birmingham.

Williamson, J.L. (1982), *Phototypesetting and Me: Survival of the Fittest*, Jovial Publications, London.

Notice that, in this style, the first line of each item (or unit) is set flush left at the margin and all other lines are indented. Thus, it appears as the opposite of the normal paragraph-style discussed in the previous section of this chapter. However, it is generated in much the same fashion—one command to define the necessary parameters and a second command to indicate the point of execution.

[ih.?] AND [ih%] AND /h COMMANDS

The command **[ih.?]** defines the size of the indent (in *picas.points* notation) for turn-over lines in "paragraphs" (or items) following any subsequent **/h** command.

Example 1

Source file contains:

```
[c24.0][f100][p10][v11]
[ih2.6]/h
Bell, M.W. (1979), [f101]Typesetting
For Beginners[f100], Creative
Press, London./h
Cross, P.A. and Double, W.P. (1978),
[f101]The Effects of Computerization
on the Printing Industry[f100],
John Brown and Sons, Birmingham./h
Williamson, J.L. (1982), [f101]Phototypesetting
and Me: Survival of the
Fittest[f100], Jovial Publications, London./h
```

Define indent-hang to be 2 picas 6 points. The command /h is necessary to cause the *first item* to conform to the pattern.

Output appears as shown near the top of this section.

To cancel the hanging-indent style, the command **[ih%]** has to be inserted. After this has been done, no further **/h** commands should be issued. However, if they are, they are interpreted as simple **/l** commands.

Note: It is *not* sufficient just to refrain from using the **/h** command to "cancel" the **[ih.?]** command, since once the indent has begun it will continue until cancelled properly with the **[ih%]** command.

The following example gives another possible use for this combination of commands. Notice this time that even the first line is involved in alignment.

Note: The symbols + . and /m appearing in this example will be explained in Chapter 9.

Example 2

Source file contains:

```
[c15.0][f100][p12][ih1.6]
+./mThis is the first item in a list, each being
marked with a ''bullet''./h
+./mNote that each ''bullet'' sits at the margin,
and the subsequent text is aligned./h
[ih%]
```

Output appears:

- This is the first item in a list, each being marked with a ''bullet''.
- Note that each ''bullet'' sits at the margin, and the subsequent text is aligned.

The success of this example relied on a knowledge of actual widths of various characters, and doesn't represent a general approach.

Further techniques concerning indentations and vertical alignment are discussed in Chapters B10–B12.

Multi-quadded lines

On occasions, we may wish to have two pieces of text appear on the same line of output, but to be positioned horizontally on that line in different ways, e.g.

Item	*Page No.*
Pointsize	17
Leading	29
Quad left in column	*Quad right in column*

While the above style can be achieved by use of the normal leading commands (**/l**, **/c** and **/r**) in conjunction with the "reverse leading" (**[vr.?]**) command, there are special quadding commands available for such situations (apart from other special quadding commands associated with the setting of tab-stops ... see Chapter B12).

/z AND /q COMMANDS

The **/z** (or *quad zero*) command operates as the **/l** command, *except that the normal leading is suppressed*, so that subsequent text will be positioned on the same line as the preceding text, *but* under the control of its own quadding command. Thus:

Example 1

Source file contains:

```
[c12.0]
I./zPointsizes/c
II./zTypefaces/c
```

Output appears:

I. Pointsizes
II. Typefaces

Source file contains:

```
[c12.0]
[f101]Item/zPage No.[f110]/r
Pointsize/z17/r
Leading/z29/r
```

Output appears:

Item	*Page No.*
Pointsize	17
Leading	29

Note that

... Text A/z... Text B/l

would cause ... Text B to overprint ... Text A.
So, usually, the command **/z** will be followed by a quadding command other than **/l** (or an indent – see Chapter B10).

The command **/q** (*quad middle*) functions in a similar way—the preceding text is set flush left and the following text is positioned on the same line (usually) by means of its own distinct quadding command (usually **/r**). However, it differs from the command **/z** in that **/z** treats the preceding and following text as *separate lines* (even if it places them on the same output line), whereas **/q** does *not cause a line-break* between the two pieces of text. If the text following the **/q** command does not fit into the remainder of the column width, then the **/q** command is ignored and normal justification takes place.

The command **/q** is especially important when the space on a line between two pieces of text set at either margin is to be filled with *leaders* (see Chapter B13), whereas it is of little use in the alignment of lists (See Chapter B10, with use of **/z**).

FORCE JUSTIFY COMMAND /j

This command forces a given output line to be justified on a particular nominated character.

Example

Source file contains:

```
[*cg8400][f100][p10][v11][c15.0]
This is a brief example to show how
lines may be forced to justify
at a particular character to avoid
unwanted hyphenation, or for other
reasons./l
```

"Natural" output appears:

This is a brief example to show how lines may be forced to justify at a particular character to avoid unwanted hyphenation, or for other reasons.

If we desire to "re-format" the same text, with the same parameters, so that the second output line ends after the word "a" (rather than the first syllable of the word "particular", then the *source file* is edited to replace the space between the words "a" and "particular" with the command **/j**, as shown below:

Source file amended to contain:

```
...
at a/jparticular character to avoid
...
```

Note: The command **/j** is used *instead of*, not as well as, the space between the words. If the space is not removed, it remains as a character so that the last character of one line or the first character of the next will be a space! In either case, the relevant line will appear unjustified.

"Modified" output now appears:

This is a brief example to show how lines may be forced to justify at a particular character to avoid unwanted hyphenation, or for other reasons.

Note: By forcing the line-ending of the second output line in this way, the line-ending of the third line was altered also because of the extra characters forced to the beginning of this line. In this case, it had the effect of removing another hyphenation.

In general, when the command **/j** is used to alter a line-ending, the endings of some or all of the subsequent lines may (but not necessarily) be changed also.

The command **/j** is useful especially to remove undesirable hyphenations, and to allow the insertion of inter-page material such as folios (page numbers) and running heads (page titles) at nominated "page breaks" (see Chapter B18).

Important Note: **/j** *cannot* be used to force *more* characters into a line than will fit naturally.

QUAD MAYBE COMMAND – /?

As shown earlier in this chapter, additional quadding commands normally produce blank lines in output. If a *macro* (see Chapter B19) is being used, it may not be clear to the operator entering the text and commands whether or not some necessary quadding command has already been included within the macro itself. To avoid this problem, it may be prudent to use the command **/?** in the macro. Its effect is

(a) to be ignored if it is immediately preceded by another quadding command (i.e. *no extra blank lines are inserted*)

(b) to be interpreted as **/l** if it is not immediately preceded by another quadding command.

QUADDING COMMANDS WITH TABS

Where tab stops have been set, special quadding commands apply to position text between tabs and then move to the next tab stop. These are discussed in Chapter B12.

CHAPTER B9
Access Codes

Each font contains 118 "visible" characters, and also implicitly has additional "invisible" characters which represent the various "fixed spaces" which may be generated. (Fixed spaces are spaces which are not affected during the justification process). As a result, each font contains more characters than there are keys on a normal terminal keyboard.

Also, some keyboard characters may not correspond to any character in some fonts and, conversely, some characters in some fonts will not correspond to any keyboard character.

Thus, some mechanism has to be employed to allow the user to access any available character despite the physical restrictions imposed by the terminal keyboard.

In this system, the following conventions are adopted:

- Wherever possible, typing any keyboard character will generate that same character (or its nearest equivalent), *provided*
  1. that character is available in the current typeface
  2. the character typed is *not* any of / or [or] or + or = or @, all of which are interpreted in a special way by the typesetting programs, nor the characters < and >.

  Thus, typing the keyboard character "asterisk" will result in the character * being output in any font in which this character exists,
  typing the keyboard character ^ (up-arrow) will generate the lower-case circumflex character if it is a member of the current font collection.
  Note: If "double quotes" are required, do *not* use the "double-quote" key on the terminal—use two single quotes (opening or closing) as the case may be.
- All other characters are referenced by an *access code* which takes one of the forms

  /x or +x

  where ***x*** is a normal alphabetic, punctuation, or numeric character.
  Thus, combination-symbols such as **/M**, **/=**, **/6**, **+c**, **+:**, **+8** will be used to access "special" characters.

  Where these characters are required, but do not exist in the current typeface, it will be necessary to issue a typeface-change command prior to using the appropriate access code.
- In the normal "text" faces, the same access code is reserved for the one character from face to face—where it does not exist, the code will be ignored.
  Thus the code **+/** will always generate *either* ✓ *or* nothing at all in these commonly-used styles.
  For faces containing a range of more specialised characters (e.g. Maths/Greek series), this consistency of access coding cannot be maintained.

For a complete list of character-availability and matching access-codes, the user should consult Appendix E in particular.

SPECIAL CHARACTERS — / AND [AND] AND + AND = AND @

Because of their special significance to JUSTIF (in connection with commands, access codes and macros), these particular six characters are *never* accessed by their single keyboard equivalent—they must be accessed by a combination code; furthermore, if they exist in the current font, they are *always* accessed by the particular access codes indicated below:

Access code // produces the character /
Access code /[produces the character [
Access code /] produces the character]
Access code /@ produces the character @
Access code /= produces the character =
Access code /+ produces the character +

Also, these access codes are never used for generating any other character.
Note: The printing characters indicated above are *not* available in all typefaces.

ACCESS CODES FOR FIXED SPACES

Another special group of "characters", which are *always* accessed by the same code, irrespective of current typeface, are the "fixed space characters". These are not true characters, but rather instructions to move the carriage of the phototypesetting machine horizontally by some fixed amount, which is not disturbed during the justification process. Since they are movements rather than characters, it follows that they are available in *all* typefaces.

The complete list of fixed spaces, togther with their individual access codes, follows:

Access code **/m** produces an "em-space", which is approximately equal to the width of the character "M"

Access code **/n** produces an "en-space", which is approximately equal in width to one-half of an "em-space", but (more importantly), in all faces *except Florentine Script*, has the same width as a numeric character (and is therefore very useful in tables for achieving alignment).

Access code **/i** produces a "thin-space", which is approximately one-third the width of an "em-space", and always has the width of certain punctuation symbols (. and , in particular) in any typeface ... again to assist is alignment.

Access code **/o** produces a "unit space", which is the smallest horizontal movement the user may request. Unit spaces may be used to produce fine adjustment between characters which appear too close.

Access code **/g** produces a "figure-space" which, for the COMPUGRAPHIC 8400, is identical to the en-space.

Access code **/f** produces a "fixed space" of approximately one-third of an "em-space".

Access code **/;** produces a "thut", which is a slightly expanded en-space.

Access code **/:** produces a "bolt", which is a slightly expanded thin-space.

In accessing these fixed spaces, upper- and lower-case characters may be used interchangeably, i.e. codes **/n** and **/N** are treated identically. (This is not true for other access codes.)

ACCESS CODES FOR DASHES

It is not uncommon that, in a single article or document, dashes of different lengths may be required. The list of common dashes and their access codes is given below:

Access code **-** (keyboard hyphen character) produces the hyphen character (-).

Access code **+ -** (keyboard combination plus-hyphen) produces the "en-dash" character (–), if it exists in the font.
The en-dash is typically used in expressing dates (1939–45), pages (pp. 234–367), etc.

Access code **_** (keyboard character underscore) produces the character "em-dash" (—).
The em-dash may be used to indicate an interruption in sentence structure, e.g.
Typesetting programs—such as JUSTIF—are available.

Access code **+ s** (keyboard combination plus-s) produces the character "minus symbol" (−), when it exists in the current typeface

Remember:

1. It is worth the effort to use these characters in the correct way to improve the appearance of the final product.
2. The complete list of access codes for all typefaces is supplied in Appendix E.

CHAPTER B10
Simple Indents

The column command **[c.?]** establishes the existence of two margins—a left margin set automatically and a right margin separated from the left margin at the distance specified in the **[c.?]** command.

An indent allows either or both of these margins to be changed temporarily.

In this chapter we will consider only the simplest indent forms—those which are most commonly used; more sophisticated techniques are explained and illustrated in the following chapter.

Thus, we need commands to indicate:

- whether the indent is to be applied to the left margin, or the right margin, or both
- the size of the relevant indent
- when to cancel (or kill) the indent.

[il.?] AND [il%] COMMANDS — LEFT INDENT

The **[il.?]** command, where *?* indicates the size of the indent in *picas.points* notation, causes a left indent to be set up at the beginning of the next output text line, and to continue until killed by an **[il%]** command (or an **[i%]** command ... see later in this chapter).

Example

Source file contains:

```
[c20.0]
This initial paragraph shows regular setting to the full
margins, so that the indent of the next paragraph
can be compared with it./l
[il5.0]The command given at the start of this paragraph
requests a left-indent of 5 picas, with no effect on the right
margin./l
[il%]Now the indent has been killed, as we see from the output./l
```

Output appears:

This initial paragraph shows regular setting to the full margins, so that the indent of the next paragraph can be compared with it.

> The command given at the start of this paragraph requests a left-indent of 5 picas, with no effect on the right margin.

Now the indent has been killed, as we see from the output.

[ir.?] AND [ir%] COMMANDS — RIGHT INDENT

These commands are identical in function and application to their **[il-]** counterparts, except that the indent created affects the right margin only.

Example

Source file contains:

```
[c20.0]
This initial paragraph will show where the regular
margins have been set, so that we may check that
the indents have worked./l
[ir5.0]This paragraph will still align on the
left margin, but will be indented by 5 picas
at the right margin./l
[ir%]The right-indent has now been cancelled, so this
final paragraph will set back to the original margins./l
```

Output appears:

This initial paragraph will show where the regular margins have been set, so that we may check that the indents have worked.

This paragraph will still align on the left margin, but will be indented by 5 picas at the right margin.

The right-indent has now been cancelled, so this final paragraph will set back to the original margins.

[ib?] AND [ib%] COMMANDS — BOTH-SIDES INDENT

Again, this pair of commands is identical in function and application to both the **[il-]** and **[ir-]** pairs of commands, except that the indent is applied to *both* the left *and* right margins.

Notes:

1. Since only *one parameter* is supplied, it follows that this command allows only a "balanced" indent, i.e. an indent of the same dimension is applied to the left and right margins.
2. If a "biased" indent is required, i.e. the left indent and the right indent are different in size, then both the **[il?]** command and the **[ir?]** command must be applied simultaneously, rather than the **[ib?]** command.

Example

Source file contains:

```
[c24.0][p10][v11]
In this example, the main text is to be
set 10 on 11 point to a measure of 24 picas.
There will also be a section representing
an extract from some other publication
(often called a ''block quote''), shown
below:/l
[ib2.0][p9][v10]
The extract will be displayed differently
from the main text, firstly by indenting
it by 2 picas on each side. Note the
command which requests this feature./l
[ib%][p10][v11]In this case, the extract has been made more distinctive by
setting it is a smaller type size (and correspondingly
smaller leading) than the main body type.
(We should have adjusted the spacing where the
leading changed.)/l
```

Output appears:

In this example, the main text is to be set 10 on 11 point to a measure of 24 picas. There will also be a section representing an extract from some other publication (often called a ''block quote''), shown below:

> The extract will be displayed differently from the main text, firstly by indenting it by 2 picas on each side. Note the command which requests this feature.

In this case, the extract has been made more distinctive by setting it is a smaller type size (and correspondingly smaller leading) than the main body type. (We should have adjusted the spacing where the leading changed.)

[i%] COMMAND — CANCEL ALL INDENTS

Any, or all, of the commands **[il?]**, **[ir?]** or **[ib?]** in effect at any time may be cancelled by the single "kill-all-indents" command **[i%]**.

Recall also, from Chapter B7, that the command **[cz]** also cancels any current indents, to restore the previously-defined column measure.

Hanging Indents

COMMANDS [ih.?] AND [ih%] AND /h

While the hanging indent commands **[ih.?]**, **[ih%]** and **/h** were introduced in Chapter B8, they are included here also for the sake of completeness.

Recall that the command **[ih.?]** establishes the size of the indent in *picas.points* notation, and the command **/h** signifies that:

- the current line is to be set flush left
- the next line should be set flush left to the *full measure*
- all subsequent lines should have a *left indent* of the size nominated in the command,

i.e. the left indent is "delayed" or "suspended" for a single output line.

The reader is referred back to Chapter B8 for examples showing some applications of this command.

Further Examples Involving Indents

In particular, some attention should be given to the construction of *lists*, such as:

1. This is a model of an item in a typical list-display, with each item of the list indicated by an Arabic numeral.
2. Notice that when the text of the item exceeds a single output line, the "turn-over" lines have to align with the first text character of the initial line of the item.
3. The general appearance resembles the "hanging indent", except for the question of alignment in the first line of each item.

While there is the basic structure of a hanging indent here, it is not really a suitable application for the **[ih.?]** command, because it requires knowledge of widths of individual characters to know "how far" to move from the numeral to the first text character in the opening line of each item.

For those already familiar with tab settings, this display cannot be easily achieved with tabs either, because of the need for turnover lines to be indented.

One technique which is useful in this situation involves the combined use of the commands **/z** and **[il.?]**, as shown in the following example.

Example

Source file contains:

```
[c20.0][f100][p9][v10]
1./z[il1.3]This is the first item of a list generated
to show the use of the ''quad zero'' command as an aid
in building indented lists./l
[il%]2./z[il1.3]Don't forget to kill the indent
before commencing each new item of the list.
Notice that alignment is correct./l
[il%]Don't forget to cancel the indent at the
end of the list also!/l
```

Output appears:

1. This is the first item of a list generated to show the use of the ''quad zero'' command as an aid in building indented lists.
2. Don't forget to kill the indent before commencing each new item of the list. Notice that alignment is correct.

Don't forget to cancel the indent at the end of the list also!

Explanatory Note:
With reference to the example above, the question can still be asked—"How was the indent value 1.3 chosen?" There are two possible ways of determining a suitable indent size:

1. The programs WIDTH and HEDFIT discussed in Chapter C2 can be used. These programs allows us to nominate the typeface and size of type to be applied to a given string of text, and they inform us of the width required to accommodate this string.
2. A working "rule of thumb" is to count each *pair* of characters as an "em" in the pointsize being used, and then convert this (by proportion) to a picas.points measure considering that in 12-point type, an "em" is 1 pica.
   Example: Consider the string "(vi) " being set in, say, 10-point type.
   Then, this string consists of 2.5 ems, i.e. 2 picas 6 points in 12-point type, i.e. 10/12 of 2 picas 6 points in 10-point type, i.e. 2 picas 1 point.
   In general, an indent estimated in this way will normally be a little large, i.e. in the example immediately above, we could try a neat 2 picas (especially) as the characters "i", "(" and ")" in the string are all "narrow" characters anyway.

Note that where there is no possibility of turnover lines, then either tabs may be used, or else you may use fixed spaces, as indicated in the following example.

Example

Source file contains:

```
[c20.0][f100][p9][v10]
1./mThis is item 1./l
2./mThis is item 2./l
3./mThis is item 3, etc./l
```

Output appears:

1. This is item 1.
2. This is item 2.
3. This is item 3, etc.

Warning:
But, if, say, alphabetic tags (e.g. (a), (b), (c), etc.) are to be employed (or Roman-numbered tags ... (i), (ii), (iii), (iv), etc.), then the method of fixed spaces shown above collapses because different alpahabetic characters have *different widths*, whereas all numeric characters have the same widths; this, then, marks a radical departure from a technique often used in typing to achieve alignment in listed material.

For alphabetic and Roman-numbered lists, the **/z-[il?]** combination is still applicable. The only problem is the selection of the size of indent to use.

Example

Source file contains:

```
[f100][p10][v11][c16.0]
This is an example of a list:/l
(i)/z[il2.0]A suitable indent is chosen to ensure
that the first text character of the item
clears the Roman-numbered tag attached to that item./l
[il%](ii)/z[il2.0]With this technique, turnover lines
will be indented correctly also./l
[il%](iii)/z[il2.0]Don't forget to cancel the indent
after each item has been inserted./l
[il%][va4]Don't forget to kill the indent after the
last item too!/l
```

Output appears:

This is an example of a list:

(i) A suitable indent is chosen to ensure that the first text character of the item clears the Roman-numbered tag attached to that item.

(ii) With this technique, turnover lines will be indented correctly also.

(iii) Don't forget to cancel the indent after each item has been inserted.

Don't forget to kill the indent after the last item too!

While the above techniques will probably be adequate for most common applications, there are circumstances for which more sophisticated routines are required, or where these basic techniques prove cumbersome. The techniques demonstrated in the following chapter, and also Chapter B12, may assist in such situations.

CHAPTER B11
More Complex Indents

The commands of this chapter are less less likely to be used regularly by many users, although there will be particular applications for which they will find extensive use.

The "picture-notch" group

In the commands **[il*?*]**, **[ir*?*]**, and **[ib*?*]**given that the command is issued at the beginning of a new "unit", the indent is applied *immediately* and continues until cancelled by the insertion of the appropriate cancelling command or a replacement command of the same kind.

In the **[ih*?*]–/h** combination, the presence of the **/h** commands indicates to *delay* the application of the indent *for one full line* and then continue the indent until another **/h** command is encountered.

Sometimes, however, we need an indent of specified magnitude to begin at some nominated point and to continue for just some specified *depth* (expressed either in picas.points notation or in terms of a known number of lines).

Illustrations of this include:

Illustration 1

This is an example of a style, referred to as a "dropped capital", used to highlight beginnings of major sections.

The "normal" text is indented for just two lines to accommodate the large introductory capital.

Illustration 2

Sometimes we have to leave room in an article for the later insertion of a photograph or a map or a diagram, etc. which has to appear in a certain position within the text. This gap is sometimes called a "picture-notch".

The size of the photograph (etc) determines both the magnitude of the indent and the depth for which the indent is to continue. This depth is expressed in picas.points notation, which can be converted to "lines of text" if the leading of the body text is known. Here, the indent is also delayed for 2 lines.

Notice that, in both cases, the point in text at which the indent is to stop (and, in the second illustration, the point in text at which it is to begin) are *not known in advance.*

This broad category of style can be otained by use of the commands discussed below.

[hl*?1*,*?2*,*?3*] COMMAND

This command is used to generate a "picture-notch" on the *left* side of the current column. This picture notch could be filled *at a later time* (as in Illustration 2 above) with a map, diagram, photograph, etc., or *immediately* with a character (or characters), such as the large-sized character used in Illustration 1 above.

In this command,

- the first parameter *?1* signifies the *size of the required left-indent* in the *picas.points* notation.
- the second parameter *?2* indicates the *number of full lines* to be set before beginning the left-indent, i.e. it is the "delay factor" in lines.
  A value of zero is taken to mean to begin the left-indent immediately (as in Illustration 1 above).
- the third parameter *?3* denotes the *number of lines* for which the indent is to continue.
  This time, a value of zero is used to indicate that the left-indent is to continue until an appropriate "cancel"-command is met (see later in this chapter).

Thus,

[hl2.6,2,5] indicates 2 full lines, then 5 lines with a left-indent of 2 picas 6 points, then full lines again.

[hl5.3,0,2] indicates an immediate left-indent of 5 picas 3 points, to be cancelled after it has been in effect for 2 lines.

[hl2.0,,6] is interpreted as **[hl2.0,0,6]**

[hl3.3] is identical to *both* **[hl3.3,0,0]** and **[il3.3]**.

[hr*?1,?2,?3*] AND [hb*?1,?2,?3*] COMMANDS

These are identical in function and application to the **[hl***?1,?2,?3***]** command, except that **[hr***?1,?2,?3***]** causes a right-indent to be produced, and **[hb***?1,?2,?3***]** causes a both-sides indent.

Returning to the opening illustrations, the source files used to produce these (using the "picture-notch" commands) are:

(a)
```
[f100][p11][v12][c16.0]
[p28]T/z[p11][vr12][hl1.9,0,2]his is an example of
a style, referred to as a ''dropped capital'', used to highlight
beginnings of major sections./l
```

(b)
```
[f100][p11][v12][c16.0]
[hr6.0,2,3]Sometimes we have to leave room
in an article for the later insertion
of a photograph or a map or a
diagram, etc. which has to appear in a certain position
within the text. This gap is sometimes called a ''picture-notch''./l
```

Notes on (a):/l

1. Notice how the "dropped capital" has been inserted with the use of the **[va***?***]** and **[vr***?***]** commands—effectively "filling in" the "picture-notch".
2. The size of the indent chosen to accommodate the "dropped capital" has been done here on a trial-and-error basis.

[hl%] AND [hr%] AND [hb%] COMMANDS

These three commands kill the corresponding "picture-notch" indents. However, there are other commands which also cancel or replace the "picture-notch" group, and a complete table is provided below:

Command	*Cancel-Commands*
[hl*?1,?2,?3***]**	**[hl%]** **[hb%]** **[i%]** **[cz]** Another **[hl***?1,?2,?3***]** command Another **[hb***?1,?2,?3***]** command
[hr*?1,?2,?3***]**	**[hr%]** **[hb%]** **[i%]** **[cz]** Another **[hr***?1,?2,?3***]** command Another **[hb***?1,?2,?3***]** command
[hb*?1,?2,?3***]**	**[hb%]** **[i%]** **[cz]** **[hl***?1,?2,?3***]** affects left-indent only **[hr***?1,?2,?3***]** affects right-indent only **[hl%]** kills left-indent only **[hr%]** kills right-indent only

Multiple-parameter sets

Where a number of "notches" are involved, all of the commands, **[hl-]**, **[hr-]** and **[hb-]**, accept up to *six sets of parameter-triples* (*?1,?2,?3*), with each set taking effect once the preceding set has been executed. The sets of three values are separated from each other by semicolons, as shown below:

Example

Source file contains:

```
[f100][p10][v11][c16.0]
[hl2.0,1,3;3.6,2,2] This example illustrates how a sequence
of left picture notches may be generated
from a single command, in which a sequence
of parameter triples are supplied.
Once the first set of parameters has been executed,
the second set is applied, and so on until
all the parameter sets have been used, or the text
is exhausted./l
```

Output appears:

This example illustrates how a sequence of left picture notches may be generated from a single command, in which a sequence of parameter triples are supplied. Once the first set of parameters has been executed, the second set is applied, and so on until all the parameter sets have been used, or the text is exhausted.

If the depth of the notch is known in terms of lines, then the **[hl–]**, **[hr–]**, **[hb–]** commands are adequate; where the depth is known in picas.points terms then, since we generally know the leading applied to the text, the same commands still suffice, since division of the picas.points measure by the leading-value yields the number of lines.

However, in circumstances where the leading may change from one revision of the document to another (or for those whose arithmetic is suspect!), there are corresponding commands which allow the depth to be specified in the picas.points notation rather than the "number of lines" form.

But, this group of "notch" commands *begin immediately*—a "number-of-lines"-delay *cannot* be specified.

[ll*?1*,*?2*] AND [lr*?1*,*?2*] AND [lb*?1*,*?2*] COMMANDS

In this group of commands,

- the parameter *?1* specifies the *size of the indent*—in picas.points notation
- the parameter *?2* specifies the *depth* (in picas.points notation) for which the indent is to continue.

Example

Source file contains:

```
[f100][p10][v11][c16.0]
[ll4.0,3.6]
The introductory command calls for a left-indent
to be applied immediately, and to continue for a
depth of 3 picas and 6 points (i.e. 42 points).
Since the leading for the text has been set at 11-point,
we expect the indent to be cancelled after 4 lines
have been set, since it is only then that the
specified depth of 42 points is exceeded./l
```

Output appears:

The introductory command calls for a left-indent to be applied immediately, and to continue for a depth of 3 picas and 6 points (i.e. 42 points). Since the leading for the text has been set at 11-point, we expect the indent to be cancelled after 4 lines have been set, since it is only then that the specified depth of 42 points is exceeded.

[ll%] AND [lr%] AND [lb%] COMMANDS

The "cancel-indent" commands are applied as for the corresponding "picture-notch" commands discussed earlier in this chapter.

The "skew" group

"Skews" are"notches" in which the size of the indent is not constant, but varies from line to line in a regular fashion.

As usual, the command may be applied to the left margin, or the right margin, or to both margins simultaneously.

The major difference is the variation in size of indent, which ranges either from zero to a maximum specified value (for so-called "positive" skews) or from a maximum specified value to zero (for "negative" skews). The increment or decrement in the indent-value from line to line is calculated automatically by the system once the *depth* of the skew has been nominated.

POSITIVE SKEW COMMANDS — [sl*?1,?2*] AND [sr*?1,?2*] AND [sb*?1,?2*]

In a "positive skew", the first line is set to the full measure, and then an indent of *increasing* size is applied to successive lines over a nominated depth *?2*, until the final line in this range is indented by the maximum amount *?1*, after which the indent is automatically killed, and setting is resumed at the full measure.

Both *?1* and *?2* are specified in the *picas.points* notation.

A left-indent of this kind is applied by the **[sl]** form of this group of commands, a right-indent by the **[sr]** form, and a both-sides-indent by the **[sb]** form, as shown in the example below.

Example

Source file contains:

```
[c20.0][f100][p10][v11]
[sl8.6,7.4]In this initial section, we are
applying a progressive left indent, beginning at
zero and increasing to 8 picas 6 points in a depth
of 7 picas 4 points.
Since the leading is 11 point and the depth is 7 picas 4 points
(i.e. 88 points), we expect the full indent to be reached in
8 lines of text, after which full measure should be resumed.
This means that the indent should increase from 0 to
8 picas 6 points in 8 lines, so that the indent should
increase by a little over 1 pica each line./l
[sr3.6,2.9]Now a demonstration of a positive
right skew is presented, with the size of the skew
increasing to 3 pica 6 points over a depth of
2 picas 9 points (i.e. 33 points, or 3 lines in
11-point leading), after which the indent is
automatically cancelled./l
[sb5.0,4.7]Finally, we have an example
of what a good diet can do, if we stick to it!
Here, there is a balanced both-sides indent,
increasing by 1 pica each side each line until a depth of
5 lines has been reached, after which full setting is restored.
This kind of setting may be useful for posters or other display work./l
```

Output appears:

In this initial section, we are applying a progressive
left indent, beginning at zero and increasing to 8
picas 6 points in a depth of 7 picas 4 points.
Since the leading is 11 point and the depth is
7 picas 4 points (i.e. 88 points), we expect
the full indent to be reached in 8 lines
of text, after which full measure
should be resumed. This means
that the indent should increase
from 0 to 8 picas 6 points in 8 lines, so that the in-
dent should increase by a little over 1 pica each line.
Now a demonstration of a positive right skew is pre-
sented, with the size of the skew increasing to 3
pica 6 points over a depth of 2 picas 9 points
(i.e. 33 points, or 3 lines in 11-point lead-
ing), after which the indent is automatically cancelled.
Finally, we have an example of what a good diet can
do, if we stick to it! Here, there is a balanced
both-sides indent, increasing by 1 pica each
side each line until a depth of 5 lines
has been reached, after which
full setting is restored.
This kind of setting may be useful for posters or other
display work.

NEGATIVE SKEWS — [sl-*?1*,*?2*] AND [sr-*?1*,*?2*] AND [sb-*?1*,*?2*]

For a "negative skew", the next text line is indented (left or right or both-sides, according to the command) by the amount *?1* (in picas.points notation), and successive lines are indented by a *decreasing* amount over the nominated depth *?2*, so that the indent "fades away" uniformly.

Note: The negative skew is indicated by the presence of a hyphen in the command, preceding the first parameter *?1*.

Example

Source file contains:

```
[c20.0][f100][p10][v11]
[sl-4.0,3.8]This example begins with a demonstration
of a negative left skew, i.e. the skew reduces from its
maximum value to a zero value over a nominated depth.
In this case, the indent reduces from 4 picas to nil over
a depth of 3 picas 8 points, which translates to 4 lines in 11-point leading./l
[sr-4.0,3.8]Here is a similar arrangement, except that the
skew indent has been applied to the right side of the text.
The values used are as for the previous paragraph.
So, we expect that the indent will end of its own accord
after 4 lines have been indented, and setting to the
full measure will be resumed./l
[sb-9.0,8.3]Here is an example of building
pyramids from the top downwards! Also, if we were to
follow this section with a similar section involving
a positive both-sides skew, we would be presenting text
organized within a diamond.
For a very large ''dropped capital'', for some
characters, the negative left skew may be preferred
to the ''hanging'' style./l
```

Output appears:

This example begins with a demonstration of a negative left skew, i.e. the skew reduces from its maximum value to a zero value over a nominated depth. In this case, the indent reduces from 4 picas to nil over a depth of 3 picas 8 points, which translates to 4 lines in 11-point leading.
Here is a similar arrangement, except that the skew indent has been applied to the right side of the text. The values used are as for the previous paragraph. So, we expect that the indent will end of its own accord after 4 lines have been indented, and setting to the full measure will be resumed.

Here
is an ex-
ample of build-
ing pyramids from
the top downwards! Also,
if we were to follow this section
with a similar section involving a po-
sitive both-sides skew, we would be pres-
enting text organized within a diamond. For a very large "dropped capital", for some characters, the negative left skew may be preferred to the "hanging" style.

COMMANDS TO CANCEL SKEWS — [sl%] AND [sr%] AND [sb%] AND [i%]

The commands **[sl***?1,?2***]** and **[sl-***?1,?2***]** are both cancelled by the command **[sl%]**, as is the *left part only* of a skew set by either **[sb***?1,?2***]** or **[sb-***?1,?2***]**. The command **[i%]** will also cancel a left skew set by any of these commands.

The commands **[sr***?1,?2***]** and **[sr-***?1,?2***]** are both cancelled by the command **[sr%]**, as is the *right part only* of a skew set by either **[sb***?1,?2***]** or **[sb-***?1,?2***]**. The command **[i%]** will also cancel a right skew set by any of these commands.

The commands **[sb***?1,?2***]** and **[sb-***?1,?2***]** are both cancelled by the command **[sb%]** and by **[i%]**. The left and right parts of these indents may be cancelled in isolation from the other as explained above.

Defining an Indent Position, Without Knowing the Size of Indent

There are occasions when the indent can most easily be defined in terms of the position within text, rather than in the normal picas.points convention.

For example, suppose we wish to align items of a list following a text string whose content, but not length, is known:

Here is a text string: Item 1
Item 2
Item 3
Item 4

Another illustration involves altering the *right margin* to fit to a text string of known content but unknown length:

The text is not to exceed the length of this line.
Now that the position of the end of the line has been marked, we can insert as much text as we wish, and it will justify on the marked position, until killed.

COMMANDS [it] AND [ij?] AND [ik?] AND [ij%] AND [ik%]

Any one of 9 different positions within the current measure can be "marked" while setting, and left or rights indents can then be made to refer to the "marks". Each marked position is given a number in the range 09, and this number is referred to in the indent being called.

To "mark" a position for subsequent indenting, use the command **[it?]**, where *?* is a number in the range 19, so that 9 distinct positions can be marked for later referral.

To use a previously-marked position as the marker for a *left-indent*, invoke the command **[ij?]**, where *?* is the same number as was used in marking the spot in text.

To use a previously-marked position as the marker for a *right-indent*, use the corresponding command **[ik?]**.

Indents set with this pair of commands are cancelled by the commands **[ij%]** and **[ik%]** respectively.

Example

Source file contains:

```
[c20.0]
Within text,[it1] we may mark positions
which are to act as boundaries for indented material./l
[ij1]Here is a left indent at a marked point./l
Note that the indent persists until cancelled by
the appropriate command./l
[ij%]
This time, we wish to force material to be indented to match
of a line of unknown length./l
This is the ''marker line''.[it2]/l
[ik2]Now we are forcing all subsequent material
to be indented on the right so that all the text
aligns with the ''marker line''./l
[ik%]
```

Output appears:

Within text, we may mark positions which are to act as boundaries for indented material.

> Here is a left indent at a marked point. Note that the indent persists until cancelled by the appropriate command. This time, we wish to force material to be indented to match of a line of unknown length.

This is the ''marker line''.

Now we are forcing all subsequent material to be indented on the right so that all the text aligns with the ''marker line''.

CHAPTER B12
Using Tabs

The current column (established by the latest **[c*?.?*]** command) can be subdivided by setting up "tab stops" (in a variety of ways) and then positioning text between tab stops by means of special quadding commands.

Warning:
1. Only simple tabular material may be set using the techniques shown in this chapter. Where the material to fit into any 'sub-column' will extend to more than a single line, it is generally more appropriate to employ strategies described in Chapter B18.
2. Any tab stops set up will be *in addition to* (not a repacement for) any tab stops currently in existence.

[ts*?1,?2,?3, ...*] COMMAND

Any number of positions, *up to a maximum of 64*, can be 'marked' as tab positions by means of this command, where *?1*, *?2*, ... are the distances *in the picas.points notation* of these positions *from the default left-margin*, i.e. tab stops are independent of indents.

Note that the **ts**-command does *not* cause anything to happen ... it simply establishes where the tab positions are to be.

[t%] COMMAND

This command cancels all tab stops (whether they have been set using the **ts**-command or by any other method—see later in this chapter). In light of the warning note above, it is a wise practice to precede any **ts**-command with a **[t%]** command, so that the new tab stops will then *replace* the old.

/u AND /v AND /w COMMANDS

When setting material using tab stops, it is necessary to use *special quadding commands*, which will position the preceding text in the current 'sub-column' and *move to the next sub-column on the same line*, whereas the 'normal' quadding commands (**/l**, **/c** and **/r**) will cause movement to the next line. The corresponding special quadding commands are, respectively,

- **/u** which positions the preceding text flush left in the current sub-column and moves on the same line to the next available tab stop,
- **/v** which performs a centering operation of the preceding text in the current sub-column and advances along the same line to the next available tab stop, and
- **/w** which sets the preceding text flush right in the current sub-column, and moves along the same line to the next available tab stop.

Note: The text in the last sub-column to be used on any given line should be terminated with a *'normal'* quadding command, so that an advance is made to the next line, rather than to the next tab stop.

Example

Source file contains:

```
[*cg8400]
[f100][p10][v11][c24.0]
[t%]                                          Cancel any existing tab stops*
[ts4.0,10.0,17.0]                             Sets up subcolumns of widths 4, 6, 7, 7 picas resp.
[f111]Ident/uName/vAmount/wCategory[f110]/c   1st column quad left, 2nd quad centre,
Item 1/uJones/v$34.78/wLocal/c                3rd quad right, 4th quad centre.
Item 2/uJackson/v$245.60/wOverseas/c          Note normal command in last sub-column
Item 3/uSmith/v$6.21/wLocal/c
```

*This is not strictly necessary in this case, as no tab stops have previously been set in this file. However, it may be wise to develop the habit of always killing old tabs whenever new one are introduced.

Output appears:

Ident	*Name*	*Amount*	*Category*
Item 1	Jones	$34.78	Local
Item 2	Jackson	$245.60	Overseas
Item 3	Smith	$6.21	Local

Note on equal widths

For all but a few special fonts (e.g. Florentine Script), in any font the widths of all numeric characters are equal (but *do* differ from font to font). They will be the same as an en-space (/n ... See Chapter 9). This is important to know, since if the quadding command in a sub-column is either **/u** or **/v**, it may be necessary to 'build up' all entries in that sub-column to the same width to achieve correct alignment.

If a sub-column contains amounts of money, the quadding command **/w** is very useful for aligning the period (.) separating dollars and cents. (Note that the width of the period is *different* from the width of each numeric character, but is the width of a thin-space (/i).)

Example

Source file contains:

```
/+345.67 volts/l
/+/n/n3.89 volts/l                    En-spaces for 'missing' figures
```

Output appears:

```
+345.67 volts
+  3.89 volts
```

/a COMMAND

This command (which again may only be used when tab stops have been set) is a 'centering'-type command, but differs from the command **/v** in that whereas **/v** centres the preceding text *between* the last tab stop and the next, the command **/a** centres the preceding text *on* the next tab stop, remaining in the sub-column beginning on the tab stop on which the centering took place, i.e. it 'straddles' two adjacent sub-columns, remaining in the second of the two.

Example

Source file contains:

```
...
[c16.0][t%][ts3.0,8.0]
[fb]Name/aCategory[fr]/c
John/uJosephson/uFinancial/c
Tom/uBrown/uUnfinancial/c
```

Output appears:

Name		**Category**
John	Josephson	Financial
Tom	Brown	Unfinancial

SETTING TABS WITHIN THE BODY OF TEXT — /t COMMAND

On occasions, we may wish to set a tab stop at a position within text (i.e. after a particular character), without knowing how far this spot is from the left margin. An example of this is shown below.

Example

Source file contains:

```
We want alignment:/n/t1. Item 1/l
/u2. Item 2/l
/u3. Item 3/l
```

Output appears:

```
We want alignment: 1. Item 1
                   2. Item 2
                   3. Item 3
```

Warning on over-setting

If an attempt is made to place more text in a sub-column than will actually fit, then the next tab stop will be passed over, and all subsequent tab movement commands (i.e. **/u**, **/v** and **/w**) will cause movement to the next available tab position, so that there will be dispacement of sub-columns.

If the last sub-column is overset, the programme will attempt to justify the entire line.

TAB STOPS FOR EQUAL SUB-COLUMNS — [tc*?*] COMMAND

The parameter *?* in this command informs JUSTIF of the number of *columns* of equal width we wish to establish, i.e. the value 3 will cause 2 tab stops to be set, creating 3 sub-columns.

Example

Source file contains:

```
[c20.0][t%][tc4]
Column 1/uColumn 2/uColumn 3/uColumn 4/l
Next 1/uNext 2/uNext 3/uNext 4/l
```

Width 20 picas, and 4 sub-columns, so that 3 tab stops will be set, at positions 5, 10 and 15 picas

Output appears:

```
Column 1   Column 2   Column 3   Column 4
Next 1     Next 2     Next 3     Column 4
```

TAB STOPS FOR SUB-COLUMNS OF PROPORTIONAL WIDTHS — [tp*?1,?2,?3,...*] COMMAND

Where sub-columns of widths in a given ratio are to be established, the **tp**-command causes the necessary arithmetic to be performed, based on the current column width, and tab stops to be set at positions dividing the total column width in that given ratio.

For example, if the current column width is 20 picas, and we wish to set up three sub-columns whose widths are in the ratio 3:2:5, then the command **[tc***3,2,5***]** will do this, i.e. it will set tab stops at distances 6 picas and 10 picas respectively from the left margin.

Example

Source file contains:

```
[c20.0]
[t%][tp3,2,5]
6 picas wide/u4 picas/Rest of column (10 picas)/l
```

Output appears:

```
6 picas wide    4 picas    Rest of column (10 picas)
```

CHAPTER B13
Inserting Leaders

On occasions, it may be desired to use some character other than whitespace to separate words or blocks of words. Any character used for this purpose is called a *leader character*.

This general style of using characters other than whitespace for separation is commonly used in items such as contents pages of documents.

The ITPS package allows us to use *either* a system-defined leader character *or* to select any available character we like as a leader character. The following commands will demonstrate how to apply these ideas.

USING SYSTEM-DEFINED LEADER ONCE — /. COMMAND

The simplest way of inserting leaders is to use the system-defined leader character (.) and to assume that we wish to use this character instead of whitespace in just one position (after which whitespace will be used again as normal). The command **/.** is used to give this "one-shot" effect.

However, to use this command, we must recognise that the leader character must fill the space between *two* pieces of text, each of which must have its own distinct and different quadding command.

Generally, the text preceding the set of leader characters will be set flush left in the column, and the text following the set of leader characters will be set flush right in the column, i.e. one piece of text is set flush left, the other piece is set flush right (with both on the *same line*), and the intervening space is "filled", not with white space, but with leader characters. The appropriate quadding command to use after the first piece of text is the command **/q**.

Note: The command **/z** is *not* appropriate, since it positions the carriage on the left margin, but we wish the leaders to begin only where the "left text" finishes, so we want the carriage positioned after the last character of the "left text", which the command **/q** does for us.

Example

Source file contains:

```
[c20.0]
Chapter 1/q/.26/r
Chapter 34/q/.423/r
Name/q/./r          Text on right side only
/./r                Full line of leaders (no text at all)
```

Output appears:

```
Chapter 1.....26
Chapter 34.....423
Name .....
.....
```

USING BASE-LINE RULE AS LEADER ONCE — /__ COMMAND

The system provides us with an "alternative" system-defined leader, viz. the base-line rule character (__). This is particularly useful in the preparation of some office-type documents, where the base-line rule may be preferred to a "row of dots" to indicate where some information is to be entered.

The command to use is /__ instead of the regular leader-insertion command **/.** introduced above. It is used in an identical fashion, as the example below demonstrates.

Example

Source file contains:

```
[c20.0]
Left text/q/__Right text/r
Enter Name:/q/__/r
/__/r                                   Rule drawn to full measure
```

Output appears:

Left text _____ Right text

Enter Name: _____

_____

The last line of the example above shows one way of inserting horizontal rules. Another technique is given in Chapter B16.

CONTINUED USE OF SYSTEM-DEFINED LEADER CHARACTER — [xl] COMMAND

In certain situations (e.g., on occasions, setting a contents page), it may be desirable to use leaders *everywhere* instead of whitespace, without having to re-apply the style in every line.

The command **[xl]** instructs the typesetting program to insert the "standard" (i.e. system-defined) leader character instead of whitespace until the instruction **[x%]** is encountered.

Example

Source file contains:

```
[c20.0]
[xl]
Chapter 1/q9/r
Chapter 2/q36/r
Chapter 3/q54/r
[x%]
F. Jones/qSydney/r
```

Output appears:

Chapter 1 9
Chapter 2 36
Chapter 3 54
F. Jones Sydney

ONE-SHOT WHITESPACE COMMAND — /s COMMAND

When leaders are in continuous use (as in the previous example), it may occur that leaders are *not* required on occasional lines. They can be temporarily suspended for a single line by inserting the command **/s** at the appropriate point.

In effect, the combination of **[xl]** and **/s** is the reverse of the normal situation with an isolated **/.** command.

DEFINING AND USING YOUR OWN LEADER CHARACTER — COMMANDS [xd*string*] AND [xx] AND /x

Although the system has effectively two distinct leader characters (**.** and **__**), there is still the opportunity for the user to select some other character or combination of characters for JUSTIF to employ whenever leaders are called for in the appropriate way.

Defining the alternative leaders

The alternative leader character or character-combination is defined by the command **[xd***string***]**, where *string* is any character or set of characters composed of not more than six individual characters or character access codes. Thus

[xd*] defines the single character * as the alternative leader character;

[xd + .] defines the single character • (accessed by the code + .) as the alternative leader character;

[xd/n./m./n] defines the combination "en-space/dot/em-space/dot/en-space" as the alternative leader character. Note that this string is considered to consist of 5 characters ... the "two-character" access codes are treated as single characters.

Invoking the alternative leaders

As with standard leaders, the alternative leaders can be applied on either a "one-shot" or "continuous" basis. The relevant commands are:

/x for the one-shot use of alternative leader defined in the **[xd***string***]** command; and **[xx]** for continuous use of alternative leader instead of whitespace, until killed by the command **[x%]**

Example

Source file contains:

```
[c20.0]
[xd*]                                   Defines * as leader for [xx] or /x commands
Left text/qRight text/r                 Leaders only defined not requested
[xx]                                    Use leaders from now on
Left Entry No 1/qRight Entry No 1/r
Left Entry No 2/qRight Entry No 2/r
Left Entry No 3/qRight Entry No 3/r
[x%]                                    Cancel use of leaders
All/qGone/r
Once/q/xOnly/r
Gone/qAgain/r
```

Output appears:

```
Left text                                        Right text
Left Entry No 1 ********** Right Entry No 1
Left Entry No 2 ********** Right Entry No 2
Left Entry No 3 ********** Right Entry No 3
All                                                  Gone
Once ************************************** Only
Gone                                             Again
```

CHAPTER B14
Ragged-Style Setting

While it is general practice to produce justified copy, there are situations in which an unjustified or "ragged" effect is desired.

The setting of *poetry*, for example, is a simple illustration—and in this case the obvious solution is to terminate each line with a quad-left code.

But, there are other examples, too—especially examples in which the final product will not correspond to the input file on a "line-to-line" basis, but still has to have a ragged appearance such as might be generated by a typewriter.

SETTING RAGGED TEXT WITH HYPHENATION — [rl],[rc] AND [rr] COMMANDS

The command **[rr]** causes subsequent text to be set with each line beginning at the left margin, but ending at a different mark, so that the right margin is not justified, but ragged (called *ragged right* style). In effect, the justifying process to which each line is normally subjected is suspended.

This produces output is a style similar to that from a typewriter.

Note: The application of the **[rr]** command alone does *not* prevent hyphenation. As much text as possible is set on any given line, without exceeding the prescribed margins, but the text is then not justified (i.e. minimum inter-word spacing only is applied).

Example 14(a)

Source file contains:

```
[*cg8400][f100][p10][v11][c12.0]
[rr]
This example shows the style called
''ragged right'' setting, and how to
achieve this style. In this example,
we have not disallowed hyphenation./l
```

Output appears:

This example shows the style called ''ragged right'' setting, and how to achieve this style. In this example, we have not disallowed hyphenation.

In like fashion, it is possible to prevent each line justifying, but then to set that line *either* centred *or* flush right, rather than flush left as in the example above. These styles are referred to respectively as *ragged centre* and *ragged left*, and are enabled by the commands **[rc]** and **[rl]**, as shown in the following two examples:

Example 14(b)

Source file contains:

```
[*cg8400][f100][p10][v11][c12.0]
[rc]
This example shows the style called
''ragged centre'' setting, and how to
achieve this style. In this example,
we have not disallowed hyphenation./c
```

Output appears:

This example shows the style called ''ragged centre'' setting, and how to achieve this style. In this example, we have not disallowed hyphenation.

Example 14(c)

Source file contains:

```
[*cg8400][f100][p10][v11][c12.0]
[rl]
This example shows the style called
"ragged left" setting, and how to
achieve this style. In this example,
we have not disallowed hyphenation./r
```

Output appears:

This example shows the style called "ragged left" setting, and how to achieve this style. In this example, we have not disallowed hyphenation.

Some Uses for Ragged Setting

Apart from the simulation of typed material, where "ragged right" is the general style used, the following situations may often lend themselves to ragged style:

(a) Headings, especially lengthy ones, which may be set any of
ragged right (such as the sub-headings throughout this manual), or
ragged centre (such as the chapter headings in this manual), or
ragged left (such as the headings on the part-title pages of this manual).
However, in this case it is usual to *also* request that *hyphenation be disabled* (see below).

(b) Text set in relatively narrow columns, where the justification process may result in ungainly inter-word spaces.

CANCELLING RAGGED STYLE — COMMAND [r%]

Only one ragged style can be in effect at any time, so that only one cancel-type command is necessary to remove the current ragged style. This command has the form **[r%]**, and takes effect immediately.

Note: If one ragged command is in effect and another is issued, then the second overrides the first (i.e. the first is cancelled).

SETTING RAGGED TEXT WITHOUT HYPHENATION — USING COMMANDS [ya%] AND [ya]

To disable *both* justification *and* hyphenation, it is necessary to include *both* the commands

[rl] or **[rc]** or **[rr]**, to prevent justification
and **[ya%]**, to suppress hyphenation.

Note: To re-introduce hyphenation, simply insert the command **[ya]**.

Example 14(d)

Source file contains:

```
[*cg8400][f100][p10][v11][c12.0]
[rr][ya%]
This example shows the style called
"ragged right" setting, and how to
achieve this style. In this example,
we have disallowed hyphenation./c
```

Output appears:

This example shows the style called "ragged right" setting, and how to achieve this style. In this example, we have disallowed hyphenation.

CHAPTER B15
Hyphenation and Justification Commands and Parameters

Important Note:

It is *very strongly suggested* that you do *not* alter the default values of the parameters of the commands discussed in this chapter without good reason.

The ITPS typesetting system incorporates both a set of general rules for producing acceptable hyphenation-points within words and a dictionary of "exceptional" hyphenations, i.e. a collection of words which generally do not hyphenate at the correct point(s) if the general rules are applied to them.

Within the standards of the system are such items as common prefixes and suffixes, instructions concerning where to break words containing various character-patterns (e.g. it is normally suitable to hyphenate a word containing at two adjacent identical consonants between this pair of characters ... "follow" can hyphenate as "fol-low"), and instructions controlling the balance between fitting lines by hyphenating words and fitting lines by opening inter-word spacing.

Clearly the complete set of instructions governing this aspect of typesetting is quite complex, and altering some part of the routine can serious affect other parts of it—hence the introductory warning above.

ENABLING/DISABLING HYPHENATION — COMMANDS [ya] AND [ya%]

Hyphenation is automatically enabled (i.e. it is initially set "on") whenever JUSTIF is applied to a source file. Thus the command **[ya]** does *not* have to be inserted at the start of a source file if hyphenation is desired (as it usually is).

However, there may be sections within the file (e.g. with headings—see Chapter B14) within which hyphenation is not regarded as desirable. These sections should be *preceded* with the command

[ya%]

which prevents hyphenation being applied to all subsequent text.

If, later, hyphenation is to be resumed, insert the command

[ya]

to re-activate the hyphenation routines.

DISABLING HYPHENATION FOR A SINGLE WORD — [yw] AND /j COMMANDS

Notwithstanding the complex hyphenation algorithm and relatively extensive dictionary, there will be occasions when a particular hyphenation is not considered satisfactory. This undesirable hyphenation point will be noticed from even a "draft" copy—see Part C.

One solution is to enclose the badly-hyphenated word between the commands **[ya%]** and **[ya]**, but there are two other solutions which are tidier.

One solution is to *precede* the single word with the command **[yw]**, which simply disallows hyphenation for one word and then automatically re-enables hyphenation.

The other solution is to *replace* the space preceding the offending word with the command **/j**, which forces the termination of that line (but still justifies it) and hence moves the whole of the offending word to the beginning of a fresh line, thus removing the need to hyphenate it.

The command **/j** was also discussed in Chapter B8.

If it is known from previous experience that a particular word can be hyphenated if necessary when you would prefer that it never be hyphenated, whenever this word is input to a source file, the command **[yw]** could always be placed in front of the word, thus preventing any hyphenation of it.

CHOOSING YOUR OWN HYPHENATION POINT — /- COMMAND

Again, if previous experience tells you that JUSTIF hyphenates a particular word at a point different from the point which you consider is the appropriate point of the word, then insert the command **/-** at the point *you* consider is the proper position. Then, when the file is being processed, the programs suspend the normal rules for deciding on hyphenation points and, if it is necessary to hyphenate the word, it will only be hyphenated at your chosen point. (If the word falls on the line such that no hyphenation is necessary, then the "discretionary hyphen" command is ignored.)

USING HYPHENATION DICTIONARY ONLY — [yd] COMMAND

As mentioned near the beginning of this chapter, hyphenation involves the use of both a set of general rules for hyphenation and reference to a dictionary of words with indicated hyphenation points.

If desired, it is possible to suspend application of the general rules and rely solely on the dictionary entries. This is achieved by inserting the command

[yd]

into the source file.

If, at a later stage, normal hyphenation routines are to be resumed, the **[yd]** command is cancelled by inserting the general enable-hyphenation command **[ya]**.

ALTERING ESSENTIAL HYPHENATION PARAMETERS — COMMAND [yp*?1,?2,?3,?4*]

This command allows the user to alter the system standards for the following hyphenation routine controls:-

- *?1*, which denotes the minimum number of letters which must occur in a word before a hyphen may be inserted (the system default value is 2)
- *?2*, which denotes the minimum number of letters which must follow an inserted hyphen (the system default value is 2)
- *?3*, which controls the number of *successive* lines on which *system-inserted* hyphen may occur (the system default value is 3)
- *?4*, which defines the minimum number of letters a word must contain before the system is permitted to hyphenate it (the system default value is 5).

Note: If *?3* is set to the value 0, then hyphenation is turned "off", i.e. it acts as the command **[ya]**.

Special Warning: In practice, it must be recognized that the forcing of a point of justification (with command **/j**) or the alteration of a particular line-ending by alteration of system-inserted hyphens by any available means can seriously affect the line-endings (including hyphenation) of all successive lines to the end of the current paragraph. So, if you make such changes, always remember to re-check to ensure you haven't introduced more problems than you have solved!

Commands affecting the Justification Process

The justification process involves positioning as much text as possible on a line (including the first part of a hyphenated word if necessary) and then adjusting the inter-word spacing till perfect justification is achieved. There are, for aesthetic reasons, limits to both the maximum and minimum allowable inter-word spaces. In addition, some systems may allow "letter-spacing" to occur (i.e. the normal clearance space between letters of a single word may be increased) if justification cannot be achieved by inter-word space adjustment only; it is usual *not* to permit letter-spacing, as the visual result is often displeasing. ITPS has letter-spacing available, but it is automatically disabled as the default setting, with the option for the user to enable this feature if he makes the conscious decision that it is desirable in the current circumstances.

The adjustment of both inter-word spacing and letter-spacing are covered in this chapter.

CHANGING ESSENTIAL JUSTIFICATION PARAMETERS — [jp*?1,?2,?3,?4*] COMMAND

This command alters system-defined values controlling the justification routines and hyphenation/justification interaction. Again, the warning must be repeated *not* to interfere with this rather delicate balance without good reason and advice.

The system-defined parameters involved are:

?1 — minimum permissible inter-word space
(the system default value is 18)

?2 — maximum inter-word space to use before attempting hyphenation
(the system default value is 27)

?3 — maximum inter-word space to use befo introducing letter-spacing
(the system default value is 54)

?4 — maximum letter-spacing permitted
(the system default value is 0, i.e. letter spacing is disabled)

Note: Each of these parameters is expressed in *relative units* (i.e. they are point-size dependent), where there are 54 relative units to the em-space in the current size.

ALLOWING/DISALLOWING LETTER-SPACING — COMMANDS [jl*?*] AND [jl%]

Letter-spacing is normally not permitted, so that the command **[jl%]** is automatically set by this system.

However, if a user desires to allow letter-spacing, the command

[jl*?*], where *?* is expressed in *relative units*

should be inserted at the point in text from which it is required.

If it is to be disabled later, the command **[jl%]** should be inserted.

Note: Although the command **/j** is really a member of the family of justification commands, it has already been introduced and discussed both earlier in the current chapter and in Chapter B8.

CHAPTER B16
Inserting Rules

While effective use of space and pointsize can dramatically reduce the need for rules in tabular and other matter, there will remain occasions when judicious use of either or both of horizontal and vertical rules can assist the reader.

In the system as implemented at this installation, there is a command available for inserting horizontal rules and a character available for inserting vertical rules.

INSERTING HORIZONTAL RULES — [or] COMMAND

The command **[or]** causes a horizontal rule to be executed (once only) immediately following the next quad left command. The rule so inserted is placed on the base-line of the line immediately following the text set flush left, and it does *not* respond to any other 'positioning' commands (i.e. to **[va.?]** and **[vr.?]** commands), since the command invokes an operation rather than a character.

However, with a little cunning, we can overcome this problem (See Example 16(c) later).

Example 16(a)

Source file contains:

```
[f100][p10][v11][c12.0]
[or]Insert a rule after the next
quad left command./I
```

Output appears:

Insert a rule after the next quad left command.

Altering the Weight (Thickness) of the Rule

The weight of the rule applied is dependent on the pointsize currently in operation, *and* the command **[or]** *does* respond to a preceding pointsize command.

Since there are 30 available pointsizes, there are 30 different weights of rule obtainable. In the following example, the technique of changing pointsize immediately before executing the rule is shown for comparison with the preceding example.

Example 16(b)

Source file contains:

```
[f100][p10][v11][c12.0]
[or]Insert a rule after the next
quad left command.[p36]/I
```

Output appears:

Insert a rule after the next quad left command.

To force re-positioning of a horizontal rule, it is necessary to introduce some (usually extraneous) character on which the 'add-leading' or 'reverse-leading' command can act.

In this situation, a 'fixed space' character can be very useful, since it is invisible to the reader! In the next example, the horizontal rule will be positioned closer to the base-line of the preceding text line.

Example 16(c)

Source file contains:

```
[f100][p10][v11][c12.0]
Insert a rule after the next
quad left command./z
[vr8][or]/n/l
```

Output appears:

Insert a rule after the next quad left command.

Warning: Because of the action of the phototypesetting command, horizontal rules are always drawn from the *default left margin* and continue to the current right margin (as set by the column-width value), i.e. left-indents are ignored, including the effective indent when multi-column work is being produced (see Chapter B18).

Because of this, some users may prefer to employ the technique of inserting a line of leaders, using the base-line ruleas the leader character ... see Chapter B13. This allows more control with regard to add- and reverse-leading commands, and the thickness of rule can be controlled by appropriate pointsize commands. The use of leaders is demonstrated in Example 16(d).

INSERTING VERTICAL RULES — VERTICAL BAR CHARACTER

The insertion of vertical rules is less mechanical, since it involves the joining of many separate characters, each of which has to be inserted.

Many (but not all ... see Appendix E) fonts contain the 'vertical rule' character, which (when available) is accessed by the corresponding keyboard character.

The use of this character to 'draw' a vertical line involves choosing the pointsize to match the current leading (remember the line has to be continuous) and then calling up the character as many times as required to produce the line of desired length.

Some Hints:

(a) Where text is also involved, it may be useful to consult the commands **[vm.?]** and **[vp.?]** (see Chapter B18), which offer control over vertical positioning.

(b) For convenience of inserting the same character in the same horizontal position on a sequence of lines, it is useful to apply the tabbing commands discussed in Chapter B12.

(c) The vertical rule character is positioned with its lower extremity slightly below the text base-line, and its higher extremity therefore does not reach the preceding text base-line. The reversal of, say 2 points, will be useful to assist in displaying contact.

Example 16(d)

Source file contains:

```
[p10][v10][c12.0]
[ts5.6,6.6]
+__/z/__/r
Left entry 1/u[vr2]|[va2]/vRight entry 1/l
Left entry 2/u[vr2]|[va2]/vRight entry 2/l
Left entry 3/u[vr2]|[va2]/vRight entry 3/l
/u[vr2]|[va2]/v/l
[vr10]+__/z/__/r
```

Output appears:

Left entry 1	Right entry 1
Left entry 2	Right entry 2
Left entry 3	Right entry 3

CHAPTER B17
Miscellaneous Commands

This chapter is devoted to a variety of commands which may not be in common use, but can be very valuable for particular occasional situations. These commands, ideas and techniques provide the user with greater flexibility and control in achieving particular styles and effects.

(I) CREATING "NEW" TYPEFACES — COMMANDS [es.?] AND [fv]

Althought the system supports almost 80 different type styles (see Appendix A and Appendix B), it is possible for the user to "manipulate" these faces to generate pseudo-typefaces with distinctly different appearances.

To a small degree this has already been mentioned in Chapter B4 with the **[fm]** command, which "tilts" the characters of any given face by a set amount. For certain sans-serif faces, this produces a reasonable "italic form" of the family.

The **[es.?]** command permits the user to "distort" the characters of any face by increasing or decreasing the width of the character without affecting its height. The parameter *.?* is any of the allowable values for a pointsize command.

Then, if the current pointsize is, say, 10 point, the command **[es18]** will cause the subsequent text to be set with the normal 10-point height, but 18-point width (i.e. an "expanded" style). Likewise, the command **[es8]** will cause subsequent text to be set with 10-point height and only the width of the corresponding 8-point character (i.e. a "condensed" style).

Example 17(a)

Source file contains:

```
[f100][p10][c12.0]
This is a sample of normal text set in
10 point type in English Times. Below the
style will be distorted./l
Normal Heading/c
[es18]Broader Heading/c
[es8]Condensed Heading/c
This paragraph continues the
condensed style begun in the previous
heading./l
[es18]Now we switch to
an expanded style. Note that
justification is still
maintained./l
```

Output appears:

This is a sample of normal text set in 10 point type in English Times. Below the style will be distorted.

Normal Heading

Broader Heading

Condensed Heading

This paragraph continues the condensed style begun in the previous heading.

Now we switch to an expanded style. Note that justification is still maintained.

A command of the form **[es?]** is cancelled by *either* the command **[es%]**, which restores the widths of all subsequent characters to that implied by the current pointsize command *or* any subsequent pointsize command, which assumes that the widths of characters will agree with the pointsize command.

There exists also the command **[we?]**, which causes the widths to correspond to those of pointsize *?* for just the remainder of the current line of type. This would be useful if "manufactured" expanded or condensed styles were to be used for headings.

The facility also exists to provide *reverse* style faces, i.e. instead of a black character on a white background, there is produced a white character on a black background. However, it is important to note that

(i) this is performed on a "single-character" basis
(ii) not all characters will yield a successful result.

The command to effect this style is **[fv]**, and it is cancelled by the command **[fv%]**.

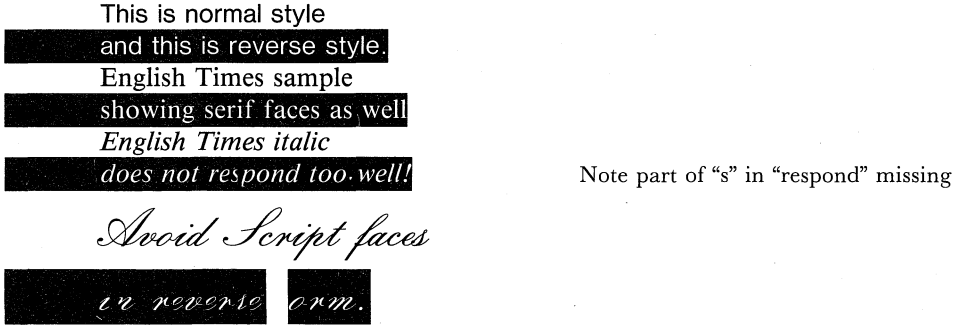
The following example demonstrates *both* how to use this command *and* some dangers associated with the "reverse-face" command.

Example 17(b)

Source file contains:

```
[p10][f520]
This is normal style/l
[fv]and this is reverse style.[fv%]/l
[f100]English Times sample/l
[fv]showing serif faces as well[fv%]/l
[f101]English Times italic/l
[fv]does not respond too well![fv%]/l
[va12][p20][f280]Avoid Script faces/l
[va12][fv]in reverse form.[fv%]/l
```

Output appears:



Note part of "s" in "respond" missing

In effect, avoid situations where a "tail" of a character can "tuck into" the "territory" of the preceding character.

(II) ALLOWING/DISALLOWING CHARACTERS TO BE PRINTED — COMMANDS [of] AND [of%]

Since the basic aim of typesetting is to produce *visible* output, it may appear strange that there should be a command which suppresses the printing of characters input to the source file! However, on occasions, the presence of this command provides a useful quick technique for assuring alignment of selected characters and for guaranteeing that a blank space is exactly the width of some nominated character string (without ever bothering to find out what the actual measure is).

Consider first the problem of a three-item list in which each item is numbered using roman numerals contained within brackets, and it is demanded that the *right* brackets should align, i.e.

(i)
(ii)
(iii)

While the necessary tab stops could be set up to perform this task (using the **/w** quadding command), another technique is to recognize the fact that we need

ii(i)
i(ii)
(iii)

where the i's outside the brackets are *not* intended to be seen.

The command **[of%]** instructs that, for all subsequent characters up to the command **[of]**, the characters should not appear, but the space these characters would normally occupy should still be inserted.

Example 17(c)

Source file contains:

```
This line shows normal left margin./l
[of%]ii[of](i)/mThis is first item/l
[of%]i[of](ii)/mThis is second item/l
(iii)/mThis is final item/
```

Output appears:

This line shows normal left margin.
(i) This is first item
(ii) This is second item
(iii) This is final item

Now consider the need to allow a space equal in width to the measure of some known text string.

Example 17(d)

Source file contains:

```
Insert name [of%]John Brown[of] directly below on following line./l
[of%]lInsert name [of]John Brown/l
```

Output appears:

Insert name John Brown directly below on following line.

Notice how the first line allows the space needed to accommodate the text string "John Brown", and the second line allows for the space needed by "Insert name".

(III) "DUMMY" END-OF-FILE MARKER — [ot] COMMAND

A very special case of the suppression of characters in the source file is the command **[ot]**, which instructs JUSTIF to ignore (i.e. not to process) all subsequent text in the input file.

Example 17(e)

Source file contains:

```
This is a sample piece of text
near the start of a long file./l
[ot]At output, this text will not appear
since it will not be processed./l
```

Output appears:

This is a sample piece of text
near the start of a long file.

The **[ot]** command is useful when, say, a trial run of just a small portion of a large file is required to either test formatting or make a minor correction.

(IV) BREAKING OUTPUT INTO SMALLER FILES — [oi] COMMAND

If the source file is very large, the output may be unmanageable — indeed, there is a maximum length of film which the take-up magazine of the phototypesetting can accept. Files producing longer output than this need to be split into a series of smaller files.

This can be done easily by inserting the command **[oi]** at suitable points throughout the .TYP file.

Suppose that the command **[oi]** is inserted at two different points in the file XYZ.TYP. We will refer to the text from the beginning of the file to the first **[oi]** command as SEGMENT 1, the body of text between the two commands as SEGMENT 2, and the text from the second of the two commands to the end of the file as SEGMENT 3. Then, when this file is processed by JUSTIF, it produces *three* output (.LST) files, instead of the normal single output file —
SEGMENT 1 (only) is placed into the file XYZ.LST (the normal name for the output file),
SEGMENT 2 is placed in a file called XYZ.L01, and
SEGMENT 3 is placed in a file called XYZ.L02.

Each of this files may then be passed separately to the phototypesetting machine.

(V) BACKSPACE COMMAND — /b

Another variation of the normal processing operations occurs when it is required to superimpose a character on the preceding character, effectively manufacturing a new (combined) character.

This may be needed when some special character is required which is not available directly as a character on any font, but can be seen to be really a combination of other existing characters, and also to generate characters with *foreign accents*, since the accents exist as separate characters in fonts where they are supplied.

The command **/b** instructs that the following character should be positioned over the preceding character, taking into account the rspective widths of the two characters involved.

As a general rule, always arrange that the character with the larger width should be set first, followed by the **/b** command, followed by the character with the smaller width.

In particular, with foreign accents, since accents generally have a narrower width than the character they relate to, the accent is inserted *after* the character.

The *exception* to this rule is where the accented character is the character "i", in which case, *two* general rules should be observed:

(i) Use the special "dotless i" character ı rather than the general i-character. Then, there will be no visual or actual interference between the "dot" of the i-character and the accent. This special character is available in all fonts with accents and is always accessed by the code +(.

(ii) Because this special character is normally *not* as wide as the accent character which is to superimpose it, it is necessary to *reverse* the order of insertion of characters in this case.

Example 17(f)
(Note that this example is shown im Mallard style, since most required characters are avilable in that font, but notice the switch to the italic form to obtain the multiplication symbol.)

Source file contains:

```
[f140]
Put an x in a box +[/b[fi]+m[fr] and carry on./l
Accents: e/b+, a/b^, ^/b+(/l
```

Output appears:

Put an x in a box ☒ and carry on.
Accents: é, â, î

CHAPTER B18
Multiple-Column Output

MARKING VERTICAL POSITIONS AND MOVING TO THEM — COMMANDS [vm.?] AND [vp.?]

The ITPS system permits us to instruct the programs to "remember" where the carriage is at any desired point in text, and to to cause the carriage to to return to that "marked" point at a later stage. In fact, we can ask it to "remember" *ten* distinct points in text!

The command to issue at a point at which it might be desired to return later has the form

[vm.?] where *?* is any integer in the range 0–9

When it is required to "move back" to a point "marked" by a **[vm.?]** command, the command to issue is

[vp.?], where *?* is the numeric value given at the time of the **[vm.?]** command

Normally, when a "move back" operation has been performed, certain other parameters (e.g. column measure, or indent-value, etc.) will have to be changed; otherwise the following text will over-print the text previously positioned at that point!

Example

Source file contains:

```
[c12.0]                                          See Note 1 below
[vm1]                                            See Note 2 below
This text is to be set in
a width of 12 picas, but we don't know how far
down the page it will set. So, if we need to
return to the beginning to set a parallel column,
we don't know the value to use in a reverse-leading
command. That is why the top of
the column has been ''marked''.
Then we can use the move-back command
to return to the chosen point at any time./l
[vm2][vp1]                                       See Note 3 below
[c27.0][il15.0]                                  See Note 4 below
Now we alter the measure to 27 picas with
a left indent of 15 picas, so that all the text which
follows will be separated by 3 picas from the
column we set initially, and will also set to
a width of 12 picas.
If this column is too short, we can move to
mark position 2, i.e. the bottom of
the left column./l
[vp2]
At the bottom!/l
```

Notes:

1. Set initial measure to 12 picas.
2. Mark position No. 1
3. Mark position No. 2 (i.e. immediately below the bottom of the left column), and move back to mark position No.1 (i.e. top of left column)
4. Reset column measure to 27 picas, with a left-indent of 15 picas, i.e. left column measure + 3 picas.

Output appears:

This text is to be set in a width of 12 picas, but we don't know how far down the page it will set. So, if we need to return to the beginning to set a parallel column, we don't know the value to use in a reverse-leading command. That is why the top of the column has been "marked". Then we can use the move-back command to return to the chosen point at any time.

Now we alter the measure to 27 picas with a left indent of 15 picas, so that all the text which follows will be separated by 3 picas from the column we set initially, and will also set to a width of 12 picas. If this column is too short, we can move to mark position 2, i.e. the bottom of the left column.

At the bottom!

This technique enables us to set column-type material in which the contents of a column may extend to more than one line (which tabbing commands cannot handle ... see Chapter B12).

The one point to remember in attempting this is to try to *set the longest column last*; otherwise it is as shown in the example above. Remember that the size of the column measure and the indent-value will determine the width in which the text will be set—the columns do not have to have equal widths.

The commands discussed below are applied to a situation which is related to the ideas above, but differs significantly in that the *depth* at which the "return to marked place" is to be performed is known, rather than the actual *word* in text at which it is to be executed.

AUTOMATIC MULTIPLE-COLUMN COMMANDS — [cm.?] AND [vg.?] AND [vc.?] AND [vb] AND [vc%]

Where multiple columns of equal width are required, and there is imposed the additional condition that columns should be of *equal depth*, the system can perform the necessary arithmetic to determine how many columns of specified width, each separated from its neighbours by a nominated horizontal space, will fit into the total width on which setting is to occur, and furthermore perform automatic returns to top of column whenever the nominated depth is exceeded. Where all columns have been filled, the system automatically repeats the pattern.

The commands involved, with their meanings and essential parameters are:

[cm.?] where *.?* denotes the total width, in *picas.points* notation, of all the columns and their separating distances.
During normal operation, if as many columns as possible as to be placed in parallel, then the value 70.0 picas should be used, since this is the maximum width of the setting area on the chemically-treated paper.
By amending the value of this parameter in this command, you can control the number of columns which may be set side-by-side.

[vg.?] where *.?* denotes the width, in *picas.points* notation, of the inter-column space, or *gutter*.

[vc.?] where *.?* defines the depth, in *points* (only) for each column,
i.e. when this depth is exceeded, a new column is automatically begun.

Example

Source file contains:

```
[f100][p10][v11]
[cm28.0][c12.0][vg3.0][vc70]
By asking for 12 pica columns and
3 pica gutters in a maximum width
of 28 picas, simple arithmetic
shows that we can place, at most, two
columns side by side. Also, since the
leading value is 11 points and the
depth to check is 70 points, there will
be 7 lines set (77 points) before
the depth is passed./l
```

Output appears:

By asking for 12 pica columns and 3 pica gutters in a maximum width of 28 picas, simple arithmetic shows that we can place, at most, two columns side by side. Also, since the leading value is 11 points and the depth to check is 70 points, there will be 7 lines set (77 points) before the depth is passed.

A new column may be *forced* near any point in text by inserting the command **[vb]** at the point. The new column will begin immediately the current line has finished setting, even though the nominated depth has not been exceeded.

More drastically, the breaking of continuous text into columns may be terminated at any time by inserting the command **[vc%]** into the file.

An application to crude pagination

By setting the value of **[c.?]** to the desired width of a page, and the value of **[vc.?]** to the desired depth of text on a page, the technique above provides a means of obtaining pages.

However, there can be certain dangers in this "automatic" approach to pagination, e.g. the positioning of headings too close to the end of a page, the possible need to insert "folios" (page numbers) and "running heads" (page titles), possible other bad page-breaks (e.g. in the middle of a table) etc. For a completely professional final product, it is often necessary to do more than simply leave the page "short". However, a preliminary "draft" version (see Chapter C1) with such a technique incorporated may assist considerably in determining where pages should begin and end.

CHAPTER B19
Macros and Arithmetic Commands

In typesetting large jobs or a large number of jobs, it can occur that the same layout is required many times. This can involve three distinct situations:

(a) A single standard ITPS command exists for producing the desired layout of type, e.g. if certain pieces of type have to be positioned on the right margin, we use the command **/r**.

(b) The layout can be achieved by a *sequence* of standard commands, e.g. **[va7][f106][p14][rr]...**

(c) No standard command exists for the purposes intended, nor may a simple sequence of standard commands suffice, e.g. to produce a list of consecutive integers 1 2 3 4 5 ... without typing each integer individually, or to generate "small" capitals letters in the current pointsize.

Even where situation *(b)* is the case, it can happen that the required sequence is lengthy and so, when it has to be repeated constantly throughout a document, it becomes tedious to have to keep inserting the command string, and the risk of error or inconsistency has to be considered.

The potential tedium in situation *(b)* can be removed, and the problem in situation *(c)* sometimes resolved, by defining and using a typesetting *macro*.

Essentially, a macro is a sequence of standard typesetting commands, with optional embedded text ... the text element may be varied from one application to another in much the same way as the value of a parameter in a standard command can be varied each time the command is applied (e.g. **[p12]** or **[p14]** or **[p20]**, etc.).

LEVEL I MACROS

At the simplest level, a macro can be considered as a user-defined command which *replaces* a string of standard commands. Such a definition is worthwhile, of course, only when the same command sequence has to be inserted many times throughout a complete job, or series of jobs; when this is the case, the macro can save many keystrokes and reduce the liklihood of inconsistency or other occasional error inherent in the task of replicating a lengthy command sequence.

Suppose that a document contains essentially two basic styles:

Style 1 — which indicates that the setting is to be 10 point on 11 point leading to the full measure, and

Style 2 — which indicates that the style is to be 9 point on 10 point leading, with a 1-pica indent on both sides.

If the document is such that we frequently have to swap from one style to the other, then we are faced with the prospect of (many times over) inserting the command strings

[p9][v10][ib1.0] to move from style 1 to style 2, and

[ib%][p10][v11] to move back to style 1 from style 2.

Even in a simple case such as this, it might be considered beneficial to reduce these command strings to more convenient "single" commands, say **[*s1]** to move to style 1 and **[*s2]** to move to style 2.

Note: Below, we shall see that the character * in such "command-names" is essential.

These "user-defined commands" can be constructed quite easily and they represent the simplest example of a typesetting macro.

To define the macro

Once the decision has been made to replace a sequence of standard commands with a macro, the *first step* is to choose a *name* for this "home-made" command. The name can consist of a maximum of 6 characters, the only illegal characters being the *space* and the *special* symbols + / = @ [ ].

In our example, for simplicity, suppose we choose to call our intended macros **s1** and **s2**.

Next, for each macro, create a file named *macroname*.MCR, i.e, in our example, the files to generate will be S1.MCR and S2.MCR. Into each file we place the command sequence which the macro is intended to replace. Thus,

S1.MCR contains **[ib%][p10][v11]**
S2.MCR contains **[p9][v10][ib1.0]**

To call (use) a defined macro

Once the macro has been defined in a .MCR file, it may be used like a "standard" command in any .TYP file, except that it is called by the command **[****macroname***]**, i.e., in our example,

```
[*s1]SEGMENT A
[*s2]SEGMENT B
[*s1]SEGMENT C
```

would give a final product in which SEGMENT A and SEGMENT C are set 10/11 to the full measure, and SEGMENT B would be set 9/10 with a 1-pica indent on each side.

Example (using macros S1 and S2 described above)

Source file contains:

```
[c20.0][f100][p10][v11
This initial text is set in the basic style, and
we now wish to shift to Style 2./l
[*S2]Note the reduction in pointsize and leading,
and the indent on both sides./l
[*S1]Now we return to the original setting,
i.e. to Style 1, with no indent and larger
pointsize and leading./l
```

Output appears:

This initial text is set in the basic style, and we now wish to shift to Style 2.

Note the reduction in pointsize and leading, and the indent on both sides.

Now we return to the original setting, i.e. to Style 1, with no indent and larger pointsize and leading.

LEVEL II MACROS — SPECIAL CHARACTER @

As an example of the construction of a more complex macro, consider the setting of a two-column table in which the entries in both columns extend to more than a single line, but the left column is always the longer—although the exact depth is variable, i.e.:

Output appears:

The left column entry is always longer than the right column entry, but the depth may vary from entry to entry.	This layout can be achieved by a repeated command sequence.
A macro to achieve this style needs to be given two pieces of variable length text at each application.	Or it may be achieved by building a suitable macro.

A "direct" approach would be to:

1. Mark top of column (i.e. current vertical position).
2. Set right-hand (shorter) column first with a suitable column measure and left indent to clear the left column *and gutter*.
3. Return to top of column, re-set column margins and insert left-hand (longer) column.

4. Repeat steps 1-3 for next entry, etc.

i.e., for two columns, each of width 12 picas, and with a 2-pica gutter, the command/text sequence would be similar to:

[vm1][c26.0][il14.0]First right column entry**/l**
[vp1][il%][c12.0]First left column entry**/l**
[vm1][c26.0][il14.0]Second right column entry**/l**
[vp1][il%][c12.0]Second left column entry**/l**
etc.

More conveniently, we need to be able to construct a macro of the form:

[vm1][c26.0][il14.0]*xxx***/l[vp1][il%][c12.0]***yyy***/l**

(where *xxx* and *yyy* represent right- and left-column text entries respectively)

which can be called for each "complete" (i.e. left and right columns) entry in the table.

The system provides for such macros. They are *defined* as for other macros (i.e. in a .MCR file), *but* they contain the special character @, which denotes that text is used at this point in execution.

For example, using the illustration above and calling the macro by the name DC (for Double-Column), the file DC.MCR would appear as

[vm1][c26.0][il14.0]@/l[vp1][il%][c12.0]@/l

To *use* this macro, and to obtain the output indicated above:

Source file contains:

```
[c29.0]][f100][p10][v11]
[*DC]This layout ... sequence.@The left column ... entry to entry.@
[*DC]Or it may ... suitable macro.@A macro to ... each application.@
```

Note: In execution, the character @ acts as a separator (or delimiter) of the different text strings to be applied in the macro.

Repeating the same macro — **[ar***?***]** *command*

Continuing the example from above, where we wish to call the same defined macro at many points in the .TYP file, there are two additional operational techniques which can be employed:

(a) Where the number of calls immediately following each other is known (in our example, 2 successive calls), we can use the command

[ar*?***]**, where *?* is the number of times to repeat the macro which *immediately follows* this command

i.e., **[ar3][*s7]** causes the macro S7 to be repeated 3 times.

Hence, referring to the example above,

Source file could contain:

```
...
[ar2][*dc]This layout ... @The left column ... @Or it ... @A macro ... @
```

(b) Alternatively—and particularly when the number of calls is not known, the insertion of an additional character @ signifies to repeat the *previous* macro call (without specifying it by name).

Again, referring to the example above,

Source file could contain:

```
[*DC]This layout ... @The left column ... @@Or it ... @A macro ... @
```

Note the additional @ symbol.

Note: The significance to JUSTIF of the symbol @ may now be appreciated. This is why special care must be taken, if the printing character @ is required in text, to use the access code /@ and not simply @.

A simple @ will be interpreted as an instruction to repeat the previous macro and, if there is no previous macro, JUSTIF will display an error message concerning macro calls; for the beginner who is not consciously using macros, this can be very confusing!

Command **[ak]**

If the definition of a macro contains reference to, say, two text strings (i.e. there are two @'s built into the definition), but it happens that on occasions only the first text string is available to be supplied, then the macro can be terminated (without supplying all of the arguments) by the command **[ak]**.

This is shown in the following example, which again uses the DC-macro defined earlier, and which simply removes the first left-column entry.

Example

Source file contains:

```
[c29.0]][f100][p10][v11]
[*DC]This layout ... sequence.@[ak]
[*DC]Or it may ... suitable macro.@A macro to ... each application.@
```

Output appears:

This layout can be achieved by a repeated command sequence.
Or it may be achieved by building a suitable macro.

A macro to achieve this style needs to be given two pieces of variable length text at each application.

Command **/?**

In Chapter B8, reference was made to this command, but no application of it was given. It is really only of use within definitions of macros to avoid possible redundant quadding commands.

Again using the DC-macro defined above, we have built the quadding command (**/l**) for each entry into the macro. But, if someone uses this macro without being familiar with the internals of the macro, it is possible that they will include the appropriate quadding command as part of the text of the entry so that, during processing, successive quadding commands will be encountered, resulting in blank lines being inserted (see Chapter B8). To avoid such possible errors, we could modify the macro by replacing each **/l** within the macro by **/?**. Then, during processing by JUSTIF,

1. If no quadding command has been inserted as part of the text of the entry, **/?** will be interpreted as **/l**.
2. If a quadding command has been inserted as part of the text of the entry, **/?** will be ignored.

An additional advantage (beyond avoiding the possible introduction of unwanted blank lines) is that it allows the quadding command for any entry to be varied (to **/c** or **/r**) simply by inserting the required quadding command as part of the text of the entry.

LEVEL III MACROS AND ARITHMETIC COMMANDS — PERFORMING CALCULATIONS

The parameters of the various commands entered into a .TYP file are held internally in the computer in storage units called *registers*. For standard commands, the registers used are well-defined by name (e.g. the register called $P holds the value of the current pointsize in hundredths of a point); in addition, the ITPS-system allows the user access to a further 260 registers (labelled A through Z and A*n* through Z*n*, where *n* takes any value in the range 1–9) also, and provides the facilities for loading a register (i.e. entering an integer value into that register), performing simple arithmetic operations on the contents of registers, inserting the contents of registers into the output file as text, and using the values in registers as arguments in commands and macros.

While the following discussion does not do justice to the whole topic, it does provide an introduction, via some simple examples, of the ideas involved.

(It is possible that a more complete treatment could form the basis for a future supplement to this manual.)

Illustration A

Suppose we require a command which generates "small capitals" (generally referred to as "small caps") — SMALL CAPITALS.

For smaller pointsizes (say, 8–12 point), it is generally sufficient simply to reduce the current pointsize by 2 (or perhaps 3) points but, for larger pointsizes, this is not an adequate working rule.

Here, it is better to observe that the size of small caps is approximately 70% of the full size, i.e. we have a formula:

required size = 70% × current size.

The following "arithmetic commands" are available to assist in this case:

[zx*R,v***]**

R is the name of a register (A-Z)

v is the integer value (or name of another register) to be placed (or loaded) into the register *R*. If a register-name is used, the integer value stored in that register is *copied* to the register being loaded.

[zm*R,v***]**

R is the name of a loaded register (A-Z)

v is the integer value (or name of another register) by which the value in *R* is to be multiplied.

The result *replaces* the original value in *R*.

[zd*R,v***]**

R is the name of a loaded register (A-Z)

v is the integer value (or name of another register) by which the value in *R* is to be divided.

The integer part of the result of this division *replaces* the original value in *R*.

Finally, if the name of a register is used as the value of a parameter in a standard command (i.e. the value in that register is used), then, in issuing the command, the register-name must be preceded by the exclamation-mark (!). Thus **[il!G]** will cause a left-indent whose magnitude (in picas) is the integer value currently stored in register G.

In our illustration, we need to:

- place the current value of the pointsize in a register (for safe-keeping, since we will want it back again later!), i.e. load the value of the "standard" register $P into, say, register C
- calculate 70% of this value, storing the result in, say, register T
- apply the normal **[p.?]** command, but using the value in register T as the value needed by the command

A suitable macro would be SC.MCR (for "small caps"), containing the following command sequence:

[zxC,$P][zxT,$P][zmT,7][zdT,1000][pr!T]

Notice that the command **[pr.?]** was used rather than **[p.?]**, because an illegal pointsize could be the result of the calculation.

The register C was used to retain the original value of the pointsize so that another macro could use this register to restore this original value.

Example

Source file contains:

```
[p10]LARGE [*sc]SMALL/l
[va12][p24]STILL[p24] [*sc]APPLIES/l
```

Output appears

LARGE SMALL

STILL APPLIES

Illustration B

Consider now the generation of a sequence of consecutive integers (perhaps to be used for folios, i.e. page numbers).

This time the following arithmetic command will be useful:

[zj*R***]**

R is the name of a register (A-Z)

This command instructs that the contents of the nominated register are to be inserted into the output (as an Arabic number), and the value in that register is then to be increased by 1.

Then, an appropriate simple macro file, NUM.MCR would contain the single arithmetic command

[zjF]

Example

Source file contains:

```
[ar10][*num]
```

Output appears:

0 1 2 3 4 5 6 7 8 9

List of other arithmetic commands

COMMAND	MEANING
[za*R***]**	The contents of *R* is written into the output file as an Arabic number.
[zw*R***]**	The contents of *R* is written into the output file in upper-case words.
[zz*R***]**	The register *R* is loaded with the particular value zero.
[zi*R***]**	The value in *R* is increased by 1.
[zp*R,v***]**	The value in *R* is increased by the value *v*, or the contents of *v* if *v* is a register-name.
[zs*R,v***]**	The value in *R* is decreased by the value *v*, or the contents of *v* if *v* is a register-name.
[zc*R,v***]**	The value in *R* is divided by the value *v*, or the contents of *v* if *v* is a register-name, and the remainder after division is stored in *R*.

There also exist four "register-testing" commands, each of which contains the name of a macro which is or is not executed, depending on the result of the test performed. Briefly, they are as follows:

[ze*R,XYZ***], [zg***R,XYZ***], [zl***R,XYZ***], [zn***R,XYZ***],**

where, in each case, *R* is the register whose value is tested, and *XYZ* is the name of the macro to be executed or ignored.

For **[ze...]**, *XYZ* is performed if the value in *R* is equal to 0; for **[zg...]**, *XYZ* is performed if the value in *R* is greater than 0; for **[zl...]**, *XYZ* is performed if the value in *R* is less than 0; for **[zn...]**, *XYZ* is performed if the value in *R* is not equal to 0.

Finally, there is the command **[vh***XYZ***]**, which causes the macro *XYZ* to be executed automatically immediately after a **[vp***?***]** command is carried out.

List of system registers with their meanings

These registers are used by JUSTIF to store values relating to specific functions. While they may not be used by the user to store values, the values held there are available to the user to use (as with $P above) in other calculations. They are all named by a single alphabetic character preceded by the symbol $.

The complete list, together with their functions, are listed below for users wishing to develop their own typesetting routines.

$B number of blocks in progress

$C current column width in eighteenths of a point (set by **[c.?]** command)

$D set-up number (relates to output device)

$E number of errors so far in file

$F current font number (set by **[f.?]** command)

$I total indent (eighteenths of a point)

$J total left indent (eighteenths of a point)

$K total right indent (eighteenths of a point)

$L number of lines thus far in file

$M amount of leading remaining in a mark point command

$N amount of leading specified in **[vc.?]** command, in eighths of a point

$P current pointsize in hundredths of a point (set by **[p.?]** command

$T total column width thus far in file

$U total leading accumulated in eighths of a point

$V current leading in hundredths of a point (set by **[v.?]** command

$W number of words thus far in file

$X pointsize resolution, set to 2 since pointsize can be requested in half-units

$Y factor declaring minimum vertical movement, set to 8 (indicating eighths of a point)

Final Example

For those interested in pursuing the construction and application of macros, the following macro is currently used in the setting of the front cover of the Prentice Computer Centre's Newsletter, to ensure that a constant line of text is always positioned exactly two lines (in 12 point leading) from the bottom of the page, irrespective of how much text has been set prior to it on this page, i.e. all the necessary variable text is entered, then the macro is called, and finally the constant line is inserted. The command sequence embedded in the macro is:

[zxP,$M][zmP,$Y][zdP,100][zsP,24][va!P]

CHAPTER B20
Typesetting from the VAX

While the principal purpose of this manual is to provide details of the ITPS typesetting package, which is available only on the KL-10 system, it should be noted that a typesetting facility exists on the VAX 11/780 machine via the TROFF program (together with the pre-processors TBL, EQN and PIC), running under EUNICE.

Formal documentation of TROFF, TBL and EQN is available, viz:

Ossanna, J.F., *NROFF/TROFF User's Manual*, Bell Laboratories Computing Science Technical Report 54, 1976.

Kernighan, B.W., *A TROFF Tutorial*, Bell Laboratories internal memorandum.

Lesk, M.E., *Typing Documents on the UNIX System: Using the -MS Macros with TROFF and NROFF*, Bell Laboratories, 1978.

Lesk, M.E., *MS Macros Reference Card*, 1978.

Lesk, M.E., *TBL — A Program to Format Tables*, Bell Laboratories Computing Science Technical Report 49, 1976.

Kernighan, B.W. and Cherry, L.L., *A System for Typesetting Mathematics*, CACM, March 1975.

Kernighan, B.W., *A System for Typesetting Mathematics — User's Guide* (2nd edition), Bell Laboratories Computing Science Technical Report 17, 1977.

Alternatively, the user may consult *UNIX User's Manual* (7th ed.), Vol. 2A.

This chapter is intended, not to replace such documentation, but merely to outline the operational procedures required to use this software and to obtain typeset output once it has been applied. By way of example, samples are provided of typical source files, together with the corresponding output.

Note: Certain aspects are still (June, 1983) in a developmental stage, e.g.

(a) it is proposed to make the facilities accessible from within the VAX/VMS operating system

(b) the method of transfer of the processed file to the final output device is currently under review

(c) full font accessibility may not be available.

Thus, such aspects of the content of this chapter are subject to change.

TROFF

TROFF is a text-processing package (which operates under EUNICE), with commands designed primarily for copy destined to be output in typeset form. Like ITPS (or any other text-processing package), it requires the user to incorporate formatting commands into the body of text to be output. In TROFF, the typical form of such commands is

either **.x**, where **x** is a one- or two-character name, often with associated parameters, (such commands appear as lines separate from normal text)

or **\x**, which may be inserted at any point in text.

In addition, there exist "macro packages" defining formatting rules and operations for specific document layouts, which can reduce the number of primitive TROFF commands needed. The MS-package, in particular, is frequently used because of the range of commonly-needed facilities it provides.

The user is referred to the listed documentation for details of the commands and macros.

PRE-PROCESSORS TBL, EQN AND PIC

TBL, EQN and PIC are TROFF pre-processors, which make available techniques for the relatively simple and convenient setting of tables (TBL), mathematical copy (EQN) and diagrammatic material (PIC), i.e. by the addition of particular commands into the source file, these auxiliary programs are called in to handle these specialised forms of setting.

In particular,

(a) commands required by TBL are included in a segment of the source file
beginning with **.TS** (for "Table Start") and
ending with **.TE** (for "Table End")

(b) commands required by EQN are included in a segment
beginning with **.EQ** and
ending with **.EN**

(c) commands required by PIC are included in a segment
beginning with **.PS** and
ending with **.PE**.

TO USE VAX TYPESETTING FACILITIES AND OBTAIN OUTPUT

(i) To run a EUNICE C-SHELL

Beginning at VAX/VMS level, the sequence is:

```
$ @bin:csh                    VAX/VMS prompt is $
%                             EUNICE prompt is %
```

TROFF and its pre-processors may now be called.

Also, at this level, the commands

MAN TROFF MAN TBL MAN EQN MAN PIC

will give brief on-line information concerning the running of TROFF and its preprocessors.

(ii) To run TROFF *(without pre-processors)*

Given a source file XYZ.TYP which contains no segments for TBL, EQN or PIC, in order to produce the processed file XYZ.LST, issue the command

```
% troff -ms xyz.typ > xyz.lst
```

(assuming the -ms macro package has been employed)

(iii) To run TROFF *(with one or more pre-processor(s))*

Since TBL, EQN and PIC are *pre*-processors, the segments relating to these programs must have their special commands translated into TROFF commands *before* TROFF can deal with these segments. Hence the appropriate EUNICE command must reference the pre-processors first, e.g.

1. `% tbl xyz1.typ | troff -ms > xyz1.lst` ... for TBL only
2. `% eqn xyz2.typ | troff -ms > xyz2.lst` ... for EQN only
3. `% pic xyz3.typ | troff -ms > xyz3.lst` ... for PIC only

In most general terms:

```
% pic xyz.typ | tbl | eqn | troff -ms > xyz.lst
```

(Where more than one pre-processor is involved, the order indicated above (i.e. PIC then TBL then EQN) is the required order.)

(iv) To leave EUNICE *after the processed file has been produced*

```
% ^D                          Control-D
$                             VAX/VMS level
```

(vi) To direct output file to phototypesetter

Since, presently, files can be sent to the phototypesetting machines only from the KL-10, it is necessary to perform a two-part operation to output processed files:

1. Transfer the .LST file from the VAX 11/780 to the KL-10, using the program NFT *from the* KL-10, as follows:

```
. r nft
NFT> xyz.lst/get:xyz.lst/node:3
User name? tscaxton
Password for node 3?
XYZ.LST <- 3::SYS$SYSDEVICE:[123987]XYZ.LST;2/BYTESIZE:8 at 18895 bps
```

2. Output normally to the phototypesetter:

```
. /cg xyz.lst
```

SAMPLE TROFF INPUT FILES

Note: The output corresponding to these input files are displayed on the following pages.

Example 1 (TROFF only)

```
.ll 24                                  Set line length to 24 ems in current pointsize
.vs 11p                                 Set leading to 11 point
.ce                                     Centre the next line
\s10\fbTROFF Sample                     10-point bold heading
.sp 4p                                  Add 4 points leading
\fr\s8TROFF\s10 has its
own distinctive set of commands.
Its standard macro packages offer convenience but
may adopt standards which may not
agree with the conventions preferred by
some users.
```

Example 2 ? (Using EQN)

```
.EQ                                     Denotes special commands (which will be recognised by
a+b over c+d = 1                        the EQN pre-processor) may be inserted.
.EN
.sp 2p
.EQ
x sup 2 + y sup 2 = z sup 2
.EN
.sp 2p
.EQ
{partial sup 2 f} over {partial x sup 2 } = x sup 2 over a sup 2 + y sup 2 over b
sup 2
.EN
.sp 2p
.EQ
sign(x)~= =~left{
rpile{1 above 0 above -1}
~lpile{if above if above if}
~lpile{x>0 above x=0 above x <0}
.EN
```

Example 3 (Using TBL)

```
.ll 20
.TS
c s s
c | c | c
l | c | n.
\fbRecent Titles
\fiTitletabAuthortabPrice\fr
Computerized Methods of Setting Type<tab>M. Rule<tab>12.70
Having Fun with Typesetting Packages<tab>M. de Sade<tab>19.25
Beginner's Guide to Typesetting<tab>I. Opener<tab>4.50
.TE
```

Example 4 (Using PIC)

```
.PS
ellipse "1"
ellipse "2" with .nw at last ellipse.se
ellipse "3" with .sw at last ellipse.ne
.PE
.PS
line right 1i down 0.25i \
        then left 1i down 0.25i \
        then right 1i down 0.25i dotted
spline from start of 1st line \
        right 1i down 0.25i \
        then left 1i down 0.25i \
        then right 1i down 0.25i
.PE
.PS
circlerad = 0.15i; d = 0.5i; s = 0.3i
define tree '
{ R: $1; move to R
{ line from R to R - $4,d chop
        line from R to R + $4,-d chop }
{ move left $4 down d - circlerad; $2 }
{ move right $4 down d - circlerad; $3 }
}
'
tree(circle "+", circle "\fIa\fP",
        tree(circle "*", circle "\fIb\fP",
                circle "\fIc\fP", s), s)
.PE
```

TROFF SAMPLE

TROFF has its own distinctive set of commands for formatting output. Its standard macro packages offer convenience, but may adopt standards which do not agree with the conventions preferred by some users.

$$\frac{a+b}{c+d}=1$$

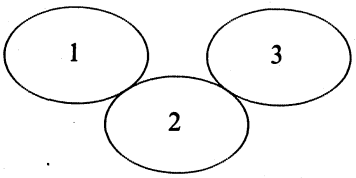
$$x^2+y^2=z^2$$

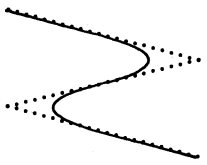
$$\frac{\partial^2 f}{\partial x^2}=\frac{x^2}{a^2}+\frac{y^2}{b^2}$$

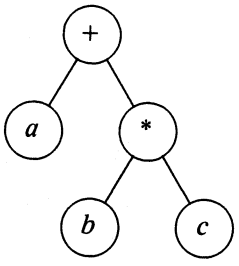
$$Sign(x) \equiv \begin{cases} 1 & \text{if } x>0 \\ 0 & \text{if } x=0 \\ -1 & \text{if } x<0 \end{cases}$$

Recent Titles

Title	*Author*	*Price*
Computerised Methods of Setting Type	M. Rule	12.70
Having Fun with Typesetting Packages	M. de Sade	19.25
Beginner's Guide to Phototypesetting	I. Opener	4.50







PART C

DETAILS OF OPERATION

CHAPTER C1
Obtaining Preliminary and Final Output

Part B of this manual was concerned primarily with the various commands which need to be inserted into the source (input) file in order that the typesetting programs will format the text on output in accordance with pre-defined specifications. In this chapter, the focus will be the means of producing the final output from the .LST file generated by the program JUSTIF.

Once the .LST file has been produced by JUSTIF, there are two possible forms of output—*either* a simulated display (on either terminal or line-printer) of the finished article *or* the genuine final product generated on the phototypesetting machine.

Since the real final output is produced on special chemically-treated paper, its creation attracts a special "materials" charge, rated (at June, 1983) at $1.50 per A4 length of paper (50¢ per 100mm length). For this reason, it is often considered desirable to produce a less expensive "mock-up" for the purposes of proof-reading for literal errors and/or inspection of general style and format before proceeding to the final-product stage. This simulated version is called a *draft copy*.

TO PRODUCE A DRAFT VERSION

The production of a draft copy involves passing the .LST file generated by JUSTIF to another program called PRELP (on the TPS: disk area), which "translates" the movement and other commands embedded in the .LST file for controlling the typesetting machine into "equivalent" commands required by a line-printer or terminal to produce an effect which, as faithfully as possible*, reflects the final product. The result of this conversion process is a file with the extension .DRF. .

* Neither a line-printer nor a terminal has the fine control characteristic of a phototypesetting machine, nor the range of characters and styles. Thus, much detail is lost in the translation.

Given that the source file XYZ.TYP has been processed by JUSTIF, yielding the file XYZ.LST, then the following two steps are carried out:

Step 1

```
.r tps:prelp
INPUT FILE FOR INTERPRETATION: xyz.lst
[Will use 209 picas ( 884 mm) or 3 A4 pages of bromide paper on the MCS8400]

END OF EXECUTION
CPU TIME: 2.73     ELAPSED TIME: 9.52
Exit

.                                          At this stage, the file XYZ.DRF has been produced
```

Note: PRELP does two things—

1. It provides information concerning the length of output which would be produced on the phototypesetter.
2. It generates a file XYZ.DRF which can be typed or printed on normal general-purpose output devices (such as a terminal or line-printer).

Step 2

To preview on a terminal:

```
.type/fortran xyz.drf
```

The simulated copy is then displayed at the terminal.

To obtain a hard-copy version from a line-printer:

```
.print/file:fortran xyz.drf
```

Special Points to Note Concerning Draft Copy

The devices on which draft files are output restrict the amount of detail which can be displayed in the draft copy. In particular:

1. Since all characters on the general-purpose output devices have the same width (unlike the characters produced by a phototypesetting machine), but the draft version still has to reflect truly the line-by-line presentation of the finished product, it follows that, whereas the final product will generally be justified, the draft copy will have a *"ragged right" appearance.*
2. The output devices do not have either the range of characters provided by the phototypesetter, nor the range of sizes. In consequence:
   (a) there is no indication of point size—*all characters have the same size*
   (b) where the true character does not belong to the character set of the "simple" output device being used (e.g. £, •, — (em-dash), etc.), some *"replacement" characters must be supplied in the draft version.*
3. There is *only one "face" available*, not the variety associated with the proper intended output device. In an attempt to provide some indication of font changes, the following conversions are always made in the draft copy:
   (i) *Italic* and *bold italic* type is shown *underlined**
   (ii) *Bold* type is shown *overprinted**
   (iii) All other typeface changes are *not* catered for.

   * With video-display terminals, these may be difficult to see.
4. The general-purpose output devices do not have a "reverse-leading" capability, so PRELP interprets *both* **[va.?]** *and* **[vr.?]** commands as *forward* advances. Thus,

   (i) a list which may appear in the final product as

   (a) The first item
   of a list of
   items.
   (b) The second
   item of the
   list.

   may appear in the draft copy as

   (a)
   The first item
   of a list of
   items.
   (b)
   The second
   item of the
   list.

   (ii) Similarly, when a superior number has been manufactured with a reverse-leading/pointsize-change combination, the draft representation of

   ... superior[36] numbers ...

   will be

   ... superior
   36
   numbers ...

While the list of limitations above may suggest that the draft version is only a poor reflection of the finished product, it must also be appreciated that the draft is comparatively inexpensive to produce (i.e. a draft copy + a final version is less expensive than an incorrect "final" version + a corrected "final" version), *and* it does allow the following valuable checks to be made:

1. Normal text characters do correspond on all output devices, so regular spelling checking can be carried out.
2. Crude format-checking can be performed—
   (i) Most common typeface changes (italic and bold) are highlighted
   (ii) Many (but not all) applications of added vertical whitespace will be shown. Fine additions (less than half a line in the current leading) are *not* treated because line-printers/terminals operate in "whole-lines" only. Even reverse-leading applications are noticeable (as shown above) although they appear as added-leading applications.

(iii) Actual line-endings are factual so, for example, bad word-breaks, column-breaks, etc are displayed.

(iv) Indents are shown to exist, even though the magnitude of the indent is not faithfully reflected. (Thus, if the command **/l** is used instead of the command **/p** and an **[ip-]** command is present with a non-zero first parameter, then the following line will not have the normal start-of-paragraph indent.) *But*, with tabular material, the internal tab-positions of the output device are called into play, with the result that PRELP *may* produce a very "mis-aligned" display, which may still be correct in the final version. Also, large indents are simply indicated by the characters >>> appearing at the left of the line, to avoid possible wrap-around of lines which can make proof-reading difficult.

It is recommended that, especially until you have sufficient experience with the programs and devices (and oftentimes even if you are experienced!), it is a good practice to produce draft copies of the entire job (document, chapter, etc) *and* "proper" output of a small sample of the job.

TO PRODUCE FINAL COPY

After examination of a draft version, and subsequent possible editing of the .TYP file and re-submission of the amended source file to the program JUSTIF to produce a modified version of the .LST file (repeating the overall process as often as needed), the stage is reached where the final product can be produced in the confidence that there are no text or format errors.

To pass the .LST file to the phototypesetting machine, there are either of two commands which may be used. They are:

```
./cg xyz.lst
```

and

```
./cgu xyz.lst
```

Both pass the file to the phototypesetting machine for processing, the only differences being *when* the job is performed and the *cost* of the processing.

The regular command is **/cg**; the command **/cgu** is a request to have the job treated as *urgent*, in which case the job is processed with as little delay as possible *but* an extra charge is levied for the preferential treatment requested.

Rules of Operation of Phototypesetter

As at June, 1983, the following rules of operation apply:
There are two "queues" leading to the machine—

- the "regular"(or MCS) queue, containing files passed by the **/cg** command; these are processed on, at worst, a 24-hour turnaround basis and attract the basic charge rate of $1.50 per A4 length of paper. If the combined length of jobs in the MCS queue exceeds 20 ft, then processing of this queue begins and the resulting output may be collected from the usual output-collection points. If this queue has not been processed prior to 5 pm on a normal working-day (because of lack of jobs), then it will be processed at that time. It is also cleared before the operators cease duty.
- the "urgent" queue, containing files passed by the **/cgu** command; these are processed *as soon as possible* and are charged at the normal rate *plus* $4.50 *per file*. This surcharge is imposed to cover the cost of wastage of bromide paper which has to occur with each processing run and additional operating overheads. With a normal queue of jobs, this wastage and other costs is shared by all jobs in the queue; but with a "single-job" situation such as occurs with jobs in the "urgent" queue, the total cost of this wastage and other overheads must be borne by the one user.

To inspect the queue and to transfer a job between queues

The COMPUGRAPHIC 8400 phototypesetting machine is known in the overall computer system as device LPT026:, so it is this queue for this device which must be inspected.

To check *all* jobs on this queue, issue the monitor-level command

```
./MCS
```

To inspect only *your own* jobs on this queue, the command is

```
./MCS/C
```

To *transfer* a job from the regular (MCS) queue, which characterises all jobs in its queue with the switch /FORMS:MCS (as you will see if you inspect the queue as shown above), to the urgent queue, which characterises all of its jobs with the switch /FORMS:URGENT, proceed as follows:

1. Determine the *jobname* of your job in the MSC queue by inspecting the queue as discussed above; part of the information returned to you is this jobname
2. Suppose that the jobname is DOCA. Then give the command

```
./MCS DOCA = /modify/forms:urgent
```

JUSTIF SWITCHES

JUSTIF accepts three switches, viz.,

/HELP, which provides information on how to apply the program, and a list of switches;

/EXIT, which causes automatic return to monitor level once the nominated file has been processed, whereas the user is normally held at JUSTIF level, i.e. there is no need for the user to give the control-C command after the file has been processed;

/LOG, which causes the generation of a *log file*, in which errors and statistics are recorded during processing. The use of this file is explained in Chapter C5.

Chapter C2
Width-Testing Programs

In some situations, it is important to know whether or not a phrase will fit into a predetermined horizontal space (e.g. whether a heading will fit into a column of prescribed width) or, conversely, to know what space is required to accommodate some given phrase (i.e. how wide to make a column, knowing the longest phrase required in that column).

The system offers two alternative programs for providing information of this kind, viz.,

TPS:WIDTH and TPS:HEDFIT

In both programs, the user nominates

- the *typeface* required
- the *pointsize* required
- the *text* required

and the program returns information concerning the width to which the given text string will set with the given typeface and pointsize parameters.

USING TPS:WIDTH

Example

```
. R TPS:WIDTH
Typesetting width calculation program
Font number? 120
Point size? 12
Text: THIS IS A SAMPLE FOR TESTING WIDTH
 22.7 ems
  22 picas +  8 points
  95.8 mm. or 3.77 inches
Text:
```

At this stage, there are three options:

either enter further text
or ^C which returns to monitor level
or ^Z which allows further testing

USING ^Z OPTION:

Each time text is supplied to WIDTH, it returns the measure, and prompts for more text (as shown above) in the assumption that pointsize and face remain constant. However, if additional tests are to be performed *but* with different pointsize and/or typeface conditions, then it is not necessary to exit (with ^C) from the program and begin again; the command ^Z returns to an earlier level (i.e. **Pointsize?** (if typed once) or **Font number?** (if typed twice)) of the WIDTH program.

Example

```
. R TPS:WIDTH
Typesetting width calculation program
Font number? 120
Point size? 12
Text: THIS IS A SAMPLE FOR TESTING WIDTH
 22.7 ems
  22 picas +  8 points
  95.8 mm. or 3.77 inches
Text: THIS IS A NEW SAMPLE
 13.3 ems
  13 picas +  3 points
  56.2 mm. or 2.21 inches
```

```
Text: ^Z
Point size? 14
Text: THIS IS A NEW SAMPLE
  13.3 ems
    15 picas +   6 points
    65.5 mm. or 2.58 inches
Text: ^Z
Pointsize? ^Z
Font number? 122
Point size? 14
Text: THIS IS A NEW SAMPLE
  13.4 ems
    15 picas +   7 points
    65.9 mm. or 2.59 inches
Text:
```

If an invalid pointsize is entered, the appropriate calculation will be made as if such a pointsize value existed. However, the program will *not* accept an invalid font number.

USING TPS:HEDFIT

Note: This program is similar in purpose and application to the program WIDTH discussed above, but was specifically written to meet the needs of "headline fitting" in magazine/newspaper applications.

The principal difference is that this time the user nominates the *measure also* and the program shows how much of the nominated text string will fit into the nominated width, as well as calculating the total width of the complete string.

Example

```
. R TPS:HEDFIT
Heading fit width calculation program
Multiple or Single Lines [Single]:        Press RETURN key for SINGLE (Default)
Column measure: 12                        i.e. 12 picas
Font number? 106
Point size? 24
That measure will contain 12 average characters
Text: SAMPLE HEADLINE
SAMPLE H/  /EADLINE                       This shows how much will fit in the column
  20 pica ems +   7 points
  87.1 mm. or 3.43 inches
Text: TEST TEXT
TEST TEXT/  /                             This shows the entire heading will fit in the column
  11 pica ems and   4 points or
  48.1 mm. or 1.89 inches
```

The following example shows the use of the "multiple-line" facility in this program.

Example

```
. R TPS:HEDFIT
Heading t width calculation program
Multiple or Single Lines [Single]: M
Font number: 106
Pointsize: 20
Measure1: 24
Measure will contain 32 average lc chars & 22 average UC chars
Measure2: 18
Measure will contain 24 average lc chars & 16 average UC chars
Measure3: 15
Measure will contain 20 average lc chars & 14 average UC chars
Measure4:                                   Press RETURN to terminate
Line1: SAMPLE HEADLINE
Line2: SAMPLE HEADLINE
Line3: SAMPLE HEADLINE
Line4:                                      Press RETURN to terminate
SAMPLE HEADLINE/  /     17 pica ems and 2 points
SAMPLE HEADLINE/  /     17 pica ems and 2 points
SAMPLE HEADLI/  /NE     17 pica ems and 2 points

Text will not fit into lines of given lengths
Total text length is 52 pica ems and 7 points
```

CHAPTER C3
Text-Processing Menu Systems — TXTMEN and JRNMEN

There exist two menu-systems containing a range of functions appropriate to users with a text-processing requirement. These two programs are called

TXTMEN a menu containing functions related to both typesetting and other word-processing tasks (notably accessing the program RUNOFF)

and JRNMEN a modified form of TXTMEN with functions suitable for typesetting only.

Users whose only use of the computer system will be in the typesetting/word-processing areas would find it *very convenient* to work with either or both of these menu-systems, since the facilities exist within them to:

- create a source file
- edit a source file
- generate a .LST file using JUSTIF
- generate a .DRF file using TPS:PRELP
- dispatch a .LST file to the phototypesetting machine
- dispatch a .DRF file to a line-printer or terminal
- perform width-checking operations, as described in Chapter C2
- perform general file-management tasks
  (e.g. TYPE, PRINT, DELETE, ARCHIVE, etc.)

In short, virtually any operation likely to be performed while setting up, processing, modifying or dispatching material destined for typesetting can be carried out from *within* these menus.

Additionally, each attempts to "protect" the user where possible by suggesting, for example, which file is intended if a file-specification is the object of a command or, again, which device is intended if a device should be specified as part of a command. This statement implies that "complete" commands do not have to be given—if a command is incomplete (i.e. the object of a command is not specified), then the user will be prompted by the menu system for the missing detail.

THE MENU SYSTEM — JRNMEN

This system is called up by the command

R JRNMEN

and it responds as indicated below:

Example

```
. R JRNMEN
        Welcome to the UQ TYPESETTING SYSTEM
             (Type ? for help to any question)
Command>
```

Note: This introductory message is displayed at the terminal only on the first occasion on each day on which the program is called up.

The possible responses to the menu-prompt (**Command>**) are:

Pressing the RETURN key, in which case the system displays a list of the commands which may be used, together with a very brief description of the function of each command (such as shown below in this chapter)

or Typing the "command" ?, as suggested in the introductory message, in which case a substantial help-message will be displayed

or Typing one of the commands from the menu.

Pressing the key RETURN will cause the system to respond with the following display:

The available commands are:

CREATE	**Create a NEW story**
EDIT	**Edit an OLD story**
HEDFIT	**Check on the fit of story heading**
JNH	**Produce a typeset story for a specified device**
DRAFT	**Draft a typeset story on the line-printer or terminal**
SET	**Send final typeset story to a specified device**
DIRECT	**List names of stories on disk**
PRINT	**Print story on the line-printer**
TYPE	**Type story on the terminal**
COPY	**Make a copy of another story**
DELETE	**Delete a specified story from storage**
EXIT	**Leave the menu system**
KJOB	**Log off from the computer system**
HELP	**Type out help information on menu system**

Notes:

1. The functions CREATE and EDIT both call up the editing program SED ... no other editor is available with this menu.
2. The function HEDFIT simply runs the program TPS:HEDFIT discussed in Chapter C2 from within the menu. Likewise, JNH runs the program JUSTIF, producing the .LST file, DRAFT runs the program TPS:PRELP, producing the .DRF file, and SET performs the same function as the command /CG, i.e. it dispatches the .LST file to the appropriate phototypesetting machine.
3. The functions DIRECT, PRINT, TYPE, COPY, DELETE, KJOB perform exactly as their monitor-level counterparts of the same name, while EXIT returns the user to monitor level, and HELP provides information.
   Note especially that you may leave the menu only via the commands KJOB or EXIT. In particular, ^C does *not* let you escape from the menu—it returns you to the general prompt **Command>**.
   Example

Command> JNH
Story to typeset: XX.TYP — Assuming this is first command
[JUSTIF Version 301 (160400)-2]
[JNHQUE - Justifying file DSK:XX.TYP]
[JHNFIN - XX DONE]
Exit

Command> SET
Story to send [XX.LST]: — Defaults to last story set by JNH if no story entered
Send to typesetting device: — Defaults to CG8400 if no device entered
[JRN Defaulting to CG8400]
Typesetting priority to be used [STD]: — STD corresponds to "normal" queue ... typing ? here will display informative message to the effect that permissible values are STD (either explicitly or by default by pressing RETURN key) or URGENT
[LPT026: XX=/Seq/232354/Limit: 6, 1 File]
Command>

THE MENU SYSTEM — TXTMEN

This menu system is similar to the one discussed above but, in differs in the following respects:

These functions are *not* available:
HEDFIT, JNH, SET, COPY

These functions are *different* in application:
CREATE, EDIT
which call up the editing program UQEDIT, not SED.

These added functions are provided:

SED, which calls up the editor of the same name
(and hence replaces the functions CREATE and EDIT from JRNMEN)

RUNOFF, which calls up the "text-processing" package RUNOFF
(which interfaces to the typesetting programs ... see below)

TYPSET, which performs a "JNH" type of function,
but with a file set up for RUNOFF.

SEND, which performs the same function as SET.

ARCHIV, RETRIE, which allow transfer of files between on-line and off-line storage areas.

Example

```
. R TXTMEN
        Text Processing Menu System
        (Type ? for help)
Menu>
```

Pressing the key RETURN will cause the system to respond with the following display:

```
The available functions are:

CREATE  Create a NEW file
EDIT    Edit an OLD file
SED     Run the SED video editor program
RUNOFF  Produce RUNOFF output for the line-printer or terminal
TYPSET  Produce a typeset file for a specified device
DRAFT   Draft a typeset file on the line-printer or terminal
SEND    Send final typeset file to a specified device
WIDTH   Find the width of typeset text
DIRECT  List names of stored files
PRINT   Print file on the line-printer
TYPE    Type file on the terminal
DELETE  Delete a specified file from storage
ARCHIV  Archive specified file offline
RETRIE  Retrieve specified file from offline
EXIT    Leave the menu system
KJOB    Log off from the computer system
HELP    Type out help information on menu system
```

RUNOFF-JUSTIF Interface

For users with files originally set up for processing by the package RUNOFF (See manual MNT-14), the menu-command TYPSET causes the RUNOFF-commands to be translated into "equivalent" JUSTIF-commands and to be passed automatically to JUSTIF, producing a .LST file.

However, since there is no provision in a word-processing package such as RUNOFF for such features as different typefaces, different pointsizes, etc., this method of treating your file adopts certain default standards and transformations, viz.,

- default output device — CG8400
- default typeface — English Times
- underlined text is converted to English Times Italic
- default pointsize — 10 point
- default leading — 12 point
- default measure — 36 picas.

Additional typesetting commands can be inserted in the .RNO file, provided they are entered in the form

[command] *or* **_/command**

i.e. the brackets or slash must be preceded by the underscore character.

The user has no real control over any of the transformations performed on a standard RUNOFF file sent for typesetting, although the .TYP file generated may be edited. However, it will contain many more commands than a regular user of JUSTIF would normally use.

CHAPTER C4
Output to APS-5 Phototypesetter

Until the installation of the COMPUGRAPHIC 8400 machine at the Prentice Computer Centre in 1983, all output had to be directed to an APS-5 phototypesetter situated at Queensland Newspapers, Bowen Hills. The facility to send output to that device still exists.

But, files prepared specifically for one machine *cannot* simply be dispatched to the other without (sometimes) *a good deal of editing* and (certainly) *re-processing by* JUSTIF.

This is because

1. the output device is different, and operates differently, so that JUSTIF has to translate your typesetting commands into operations which the individual particular machine will understand
2. the typefaces available on the different machines are different (although some may be common) and are referenced by *different font numbers* even when the face is available in both, e.g.
   (a) English Times is available on both machines, but whereas it is font number 100 on the COMPUGRAPHIC 8400 machine, it is font number 230 on the APS-5 machine
   (b) this manual could not have been set on the APS-5 machine, since Baskerville is not available on that machine
   (c) the Newton series of typefaces is available on the APS-5 machine, but not on the resident COMPUGRAPHIC 8400 machine.
3. the character-sets on any common fonts do not coincide—each APS-5 font contains 100 characters (not 118 as on the COMPUGRAPHIC 8400 machine).
4. access codes may differ for the same character on the different machines.

Where there is good reason to wish to use the APS-5 machine, you are advised to consult the Prentice Computer Centre regarding font and character availability, access codes and font numbers.

Also, the first command in the file must be

[*aps] *not* **[*cg8400]**.

TO OBTAIN OUTPUT FROM APS-5

If operating from monitor-level, give the command

```
./TYPESET XYZ.LST/NOHEAD
```

after the file XYZ.LST has been produced by JUSTIF.

If operating from menu-level (either JRNMEN or TXTMEN), the command **SET** or **SEND** (as the case may be) will request information about the output device. In either case, enter the value **APS**.

OPERATING RULES

Since output to the APS-5 machine requires the physical transportation of materials to another location, it is inevitable that turnaround time will be greater. The dispatch arrangements are:

1. The .LST file is written to magnetic tape at midnight on any working day.
2. The magnetic tape is dispatched on the following working morning to Queensland Newspapers for processing.
   (i.e. jobs submitted on Friday will be written to magnetic tape at midnight, and collected for dispatch on Monday morning.)
3. Output is collected from Queensland Newspapers on the following working day and are normally available for collection at the Prentice Computer Centre that afternoon.

With regard to *cost*, normal computer processing costs are involved, plus the "materials surcharge", which is rated at $2.00 per A4 length of page by Queensland Newspapers.

CHAPTER C5
Tracking Down Errors

When an illegal command is issued in a .TYP file, JUSTIF returns an error message to the terminal when it encounters this illegal command during processing, usually displaying some of the text around the offending command (as an aid to the user in locating the incorrect command during subsequent editing), as well as a brief comment on the illegality, e.g.

```
text ... [F161]ILLEGAL FACE
<%JUSERR:
      ?
FACE NOT AVAILABLE
      >
```

However, depending on the error, the message may not always provide a simple or meaningful comment on what is wrong (such as ILLEGAL FACE), e.g.

```
text ... /0 ...
<%JUSERR:
?
ILLEGAL COMMAND
>
```

In this case, the display of some of the surrounding text can be valuable, since the only simple method of detecting the error is:

1. To return to the .TYP file and use the text displayed to locate the section of the file where the error occurs, and then
2. To inspect very carefully all the text/commands in that section for possible errors, which could include
   illegal commands, eg. [k20] instead of [p20], /y instead of /r
   missing command indicators or incorrect access codes, e.g. + instead of /+ or /0 (which is not a valid access code), etc.

If you cannot detect the source of the error in this manner, then the techniques described below may be useful.

GENERATING .JH (ERROR/STATISTICS) FILE

When running program JUSTIF, an auxiliary file (.JH file) may be generated along with the .LST file, in which is recorded:

1. each error (if any) detected, together with the line-number of the line of the input file on which the error occurs, and a comment on the error itself (similar to the terminal display when an error is met), and
2. summary statistics on the input file, viz.,
   font number of primary typeface, number of errors, number of keystrokes, number of words, number of lines, value of primary leading.

By default, this file is not generated unless specifically requested by the user when calling the program JUSTIF, as shown below (taking the file DOCA.TYP as the input file):

```
.R JUSTIF
*DOCA/LOG                    Use the /LOG switch
```

Sometimes the .JH file so generated may assist is locating and recognizing an error in the .TYP file.

GENERATING .DMP FILE

A program TPS:DMPBIN exists which translates the contents of a .LST file into a humanly-intelligible set of instructions and text. Basically it records, step by step, the primitive instructions which the phototypesetting machine will obey—i.e. every horizontal/vertical movement (in direction and magnitude), each typeface change, each pointsize change, each word printed.

Consequently, the file generated by DMPBIN will usually be a lengthy file but, as a last resort, may help to detect errors at a point in text by displaying exactly what is happening at that point.

To generate a .DMP file, the .LST file must first be generated by JUSTIF, and then passed as the input file to DMPBIN as follows:

```
.R TPS:DMPBIN
INPUT FILE NAME: XYZ.LST
INPUT FILE NAME:^C
```

The file generated by DMPBIN will be named XYZ.DMP.

SEEKING OTHER ADVICE

Other enquiries regarding typesetting can be directed to the Prentice Computer Centre.

In particular, consulting problems can be forwarded via the program MAIL to the mailbox TEXT as follows:

```
.MAIL
No mail
Mail command: SEND
To: TEXT
Subject: CONSULTING
Message: (^Z when done)
....ENTER MESSAGE ...
```

APPENDIX A

List of Available Fonts, with Associated Font Numbers

For organizational purposes, and convenience of use, the following convention has been adopted in the numbering of the fonts:

The "normal" face of a family of faces (usually the 'medium' face) has an associated font number terminating with 0,
e.g. English Times Medium is font 100, Baskerville Medium is font 110, Souvenir Medium is font 150, Eras Medium is font 510, and so on.
Where there is only one member in the family (e.g. Clearface Contour, Goudy Extrabold, Florentine Script, the Phonetic and Mathematics/Greek fonts) this face will also have a number ending in 0.

The "medium italic" face of the family has a font number greater by 1 than the "normal" face,
e.g. English Times Italic is font 101, Baskerville Italic is font 111, Souvenir Medium Italic is font 151, etc.
Where no such member of the family exists (e.g. Eras, University, Rockwell, etc), no other face will take this number.

The "bold" face of the family has a font number greater by 2 than the "normal" face,
e.g. English Times Bold is font 102, Baskerville Bold is font 112, Souvenir Bold is font 152, Eras Bold is font 512, etc.
Where the bold form does not exist, the number is not used for any other face.

The "bold italic" form of the family (where it exists) has been given a font number greater by 3 than the "normal face", and this number is reserved for such a face only,
e.g. English Times Bold Italic is font 103, Baskerville Bold Italic is font 113, Souvenir Bold Italic is font 153; Eras Bold Italic is not available, so font number 513 is not used.

Also, since no family of faces available contains more than nine members, the first two digits of the font numbers for all members of any family are identical, e.g.
all English Times styles have numbers of the form 10?,
all Baskerville styles have numbers of the form 11?,
all Souvenir styles have numbers of the form 15?,
all Eras styles have numbers of the form 51?, and so on.

Where styles other than the four "basic" styles (Medium, Medium Italic, Bold, Bold Italic) exist, the last digit of the three-digit font number will lie in the range 4–9,
e.g. Souvenir Light is font 155, Univers Ultrabold Expanded is font 508, English Times Condensed is font 104, Helios Semibold Outline is font 524.

Note: For these "non-basic" styles, there is no regular pattern to the allocation of the final digit in the font number.

Below are listed all the available fonts, with their associated font numbers, arranged in families, in increasing font number order.

Font Number	*Face Description*
100	English Times
101	English Times Italic
102	English Times Bold Text
103	English Times Bold Italic Text
104	English Times Condensed
105	English Times Bold Display
106	English Times Bold Italic Display
110	Baskerville
111	Baskerville Italic
112	Baskerville Bold
113	Baskerville Bold Italic
120	Garth Graphic
121	Garth Graphic Italic
122	Garth Graphic Bold
123	Garth Graphic Bold Italic
130	Century Schoolbook
131	Century Schoolbook Italic
132	Century Schoolbook Bold
133	CenturyBold Italic
140	Mallard
141	Mallard Italic
142	Mallard Bold
143	Mallard Bold Italic
150	Souvenir Medium
151	Souvenir Medium Italic
152	Souvenir Bold
153	Souvenir Bold Italic
154	Souvenir Outline
155	Souvenir Light
156	Souvenir Light Italic
160	Rockwell Medium
162	Rockwell Bold
170	Serif Gothic Regular
172	Serif Gothic Bold
174	Serif Gothic Extrabold
180	Tiffany Medium
182	Tiffany Bold
190	Grouch
200	Korinna Extrabold
210	Goudy Extrabold
280	Florentine Script
300	Special Logos*

Font Number	*Face Description*
500	Univers Medium
501	Univers Medium Italic
502	Univers Bold
504	Univers Extrabold
505	Univers Light
506	Univers Light Italic
507	Univers Extrabold Expanded
508	Univers Ultrabold Expanded
510	Eras Medium
512	Eras Bold
514	Eras Demibold
515	Eras Ultrabold
516	Eras Outline
520	Helios
522	Helios Bold
524	Helios Semibold Outline
530	University Roman
532	University Bold
540	Stencil
544	Stencil Outline
550	Clearface Contour
760	Dingbats 200 (Miscellaneous No. 1)
770	Dingbats 300 (Miscellaneous No. 2)
900	Script Text (Maths/Greek No. 1)
910	Full-size Greek Text (Maths/Greek No. 2)
920	Inferior English Times Italics (Maths/Greek No. 3)
930	Inferior English Times (Maths/Greek No. 4)
940	Superior Greek and Chemistry (Maths/Greek No. 5)
950	Superior English Times Italic (Maths/Greek No. 6)
960	Inferior Greek (Maths/Greek No. 7)
970	Superior English Times (Maths/Greek No. 8)
980	Phonetic English (Phonetics No. 1)
990	International Phonetic Alphabet (Phonetics No. 2)

* At time of preparation of this manual, font 300 is reserved only for this purpose ... no characters have yet been included in this font, but arrangements are being made to include logos of special interest to particular groups on campus (e.g. University of Queensland shield, Griffith University shield, Prentice Computer Centre logo, etc).

An example of each style of type is provided in Appendix B.

APPENDIX B
Samples of All Available Type-Styles

For purposes of inspection and comparison, a common piece of text has been set in each 'text' type-style available on the COMPUGRAPHIC 8400 phototypesetter.

However, it should be carefully noted that particular styles of type are best suited to particular purposes, and the design used in the samples below will *not* display most of the typefaces to their greatest advantage.

It is recommended that, unless you have special knowledge of particular typefaces, you should experiment with small amounts of text to determine a style of type which best suits your individual requirements. Also, take care where "spectacular" display faces are involved ... there are normally used in very short "bursts", not for lengthy text pieces.

All samples have been set in 11 point on 12 point leading.

English Times:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

English Times Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

English Times Bold Text:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

English Times Bold Italic Text:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

English Times Condensed:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

English Times Bold Display:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

English Times Bold Italic Display:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Note: The distiction between "Text" and "Display" styles in English Times is that the "Text" faces should be used for point sizes up to approximately 24 point, and the corresponding "Display" faces used for all larger point sizes. A few characters have been modified in the "Display" forms to avoid noticeable distortion in the larger pointsizes. It is legal to use either in any available size, but best results will be obtained if the preceding rule is followed.

Baskerville:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Baskerville Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Baskerville Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Baskerville Bold Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Garth Graphic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Garth Graphic Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Garth Graphic Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Garth Graphic Bold Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Century Schoolbook:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Century Schoolbook Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Century Schoolbook Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Century Bold Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Mallard:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Mallard Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Mallard Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Mallard Bold Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Medium:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Medium Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Bold Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Outline:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Light:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Souvenir Light Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Rockwell Medium:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Rockwell Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Serif Gothic Regular:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Serif Gothic Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Serif Gothic Extrabold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Tiffany Medium:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Tiffany Light:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Grouch:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Korinna Extrabold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Goudy Extrabold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Florentine Script:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Eras Medium:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Eras Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Eras Demibold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Eras Ultrabold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Eras Outline:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Medium:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Medium Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Extrabold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Light:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Light Italic:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Extrabold Expanded:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Univers Ultrabold Expanded:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Helios:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Helios Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

Helios Semibold Outline:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

University Roman:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

University Bold:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

STENCIL:
THIS SHORT SAMPLE OF TEXT HAS BEEN SET TO DEMONSTRATE THE VARIOUS STYLES OF TYPE CURRENTLY AVAILABLE, AND TO ALLOW COMPARISON BETWEEN THESE STYLES.

STENCIL OUTLINE:
THIS SHORT SAMPLE OF TEXT HAS BEEN SET TO DEMONSTRATE THE VARIOUS STYLES OF TYPE CURRENTLY AVAILABLE, AND TO ALLOW COMPARISON BETWEEN THESE STYLES.

Clearface Contour:
This short sample of text has been set to demonstrate the various styles of type currently available, and to allow comparison between these styles.

APPENDIX C
List of Available Characters in Each Font

When selecting a suitable typeface for a particular job, it will be necessary to check that all characters needed are available in the chosen font. The "character-lists" below for each font may be found useful for this.

Also, where it is found necessary to use another font for just an occasional character, the lists below may help in deciding which alternate font should be used to match the common font most closely.

It is also recommended that you consult Appendix E, since this defines the required access code for any character available in a font.

English Times
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)‘?’[*]-•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ _ ‡ ► ″ § ☆ ¶ ● ÷ ◄ ′ © × ± ® –

English Times Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)‘?’[*]-•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ _ ‡ ► ″ § ☆ ¶ ● ÷ ◄ ′ © × ± ® –

English Times Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)‘?’[*]-•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ _ ‡ ► ″ § ☆ ¶ ● ÷ ◄ ′ © × ± ® –

English Times Bold Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)‘?’[*]-•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ _ ‡ ► ″ § ☆ ¶ ● ÷ ◄ ′ © × ± ® –

English Times Condensed
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)‘?’[*]-•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ _ ‡ ► ″ § ☆ ¶ ● ÷ ◄ ′ © × ± ® –

English Times Bold Display
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)‘?’[*]-•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ _ ‡ ► ″ § ☆ ¶ ● ÷ ◄ ′ © × ± ® –

English Times Bold Italic Display
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# { }| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ ‡ ► " § ☆ ¶ ● ÷ ◄ ' © × ± ® –

Baskerville
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]•/&–$
%#˜ ^ ` ⅛ ¼ ⅜ ½ ⅝ ¾ ⅞ ⅓ ⅔ ¡ ` £ ´ ¸ ¨ ¿ ¢ □ ˆ _ ˜ ıß ffi ffl ff fi fl

Baskerville Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–/&—$
%˜ ^ ` + = ¡ ■ ° £ ´ ¸ † ¨ ¿ ˆ _ ˜ ıß • ffi ffl § ÷ ff fi © fl × ® –

Baskerville Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]•/&—$
%#˜ ^ ` ⅛ ¼ ⅜ ½ ⅝ ¾ ⅞ ⅓ ⅔ ¡ ` £ ´ ¸ ¨ ¿ ¢ □ ˆ _ ˜ ıß ffi ffl ff fi fl

Baskerville Bold Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–/&—$
%˜ ^ ` + = ¡ ■ ° £ ´ ¸ † ¨ ¿ ˆ _ ˜ ıß • ffi ffl § ÷ ff fi © fl × ® –

Garth Graphic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789 ¢ □ _ / 1234567890 ®

Garth Graphic Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789 ¢ □ _ / 1234567890 ®

Garth Graphic Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789 ¢ □ _ / 1234567890 ®

Garth Graphic Bold Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼½¾⅓⅔$★✓0123456789¢□1234567890®

Century Schoolbook
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'*–•/&—$
%#˜ˆˋ+@=¡˚°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

Century Schoolbook Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'–•/&—$*
%#˜ˆˋ+@=¡˚°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

Century Schoolbook Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'*–•/&—$
%#˜ˆˋ+@=¡˚°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

Century Bold Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'*–•/&—$
%#˜ˆˋ+@=¡˚°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

Mallard
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]•/&–$
%#˜ˆˋ⅛¼⅜½⅝¾⅞⅓⅔¡˙£´ı¸¨¿¢□ˆ__˜1ßffiffflfffifl

Mallard Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–/&–$
%˜ˆˋ+ =¡˙■°£´ı¸†¨¿ˆ__˜1ß·ffiffl§÷ff'fi©fl×®–

Mallard Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]•/&–$
%#˜ˆˋ⅛¼⅜½⅝¾⅞⅓⅔¡˙£´ı¸¨¿¢□ˆ__˜1ßffiffflfffifl

Mallard Bold Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–/&—$
%˜ ^ ` + = ¡ ■ ° £ ´ ı ¸ † ¨ ¿ ˆ _ ˜ 1ß • ffiffl§ ÷ ff fi © fl × ® –

Souvenir Medium
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Souvenir Medium Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Souvenir Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Souvenir Bold Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Souvenir Outline
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Souvenir Light
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Souvenir Light Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% 1/4 1/2 3/4 1/3 2/3 $ ★ ✓ 0123456789 ¢ Ç □ __ /1234567890 ®

Rockwell Medium
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% ¼½¾⅓⅔$★✓0123456789¢¢□__/1234567890®

Rockwell Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
% ¼½¾⅓⅔$★✓0123456789¢¢□__/1234567890®

Serif Gothic Regular
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]/&—$
% $f0123456789¢¢__AEEEkMNQRsZaaefhkmnyz

Serif Gothic Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]/&—$
% $f0123456789¢¢__AEEEkMNQRsZaaefhkmnyz

Serif Gothic Extrabold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]/&—$
% $f0123456789¢¢__AEEEkMNQRsZaaefhkmnyz

Tiffany Medium
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]/&—$
% $&0123456789¢¢__ECKLNQRS Uhfhkmnrs vwxy

Tiffany Light
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]/&—$
% $&0123456789¢¢__ECKLNQRS Uhfhkmnss vwxy

Grouch
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-•/&—$
%# {}|+@=¼½¾⅓⅔■°★†✓□__❉►"§☆¶●÷◄'©×±® –

Korinna Extrabold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@=¼½¾⅓⅔■°★†✓□_‡►"§☆¶●÷◄'©×±® –

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

Florentine Script
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]¢•/&—$
#^`¼½¾⅓° st nd rd th %_ ÇD I O sVXç~ c

Univers Medium
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@=¼½¾⅓⅔■°★†✓□_‡►"§☆¶●÷◄'©×±® –

Univers Medium Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@=¼½¾⅓⅔# °★†✓□_‡►"§☆¶●÷◄'©×±® –

Univers Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@=¼½¾⅓⅔■°★†✓□_‡►"§☆¶●÷◄'©×±® –

Univers Extrabold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@=¼½¾⅓⅔■°★†✓□_‡►"§☆¶●÷◄'©×±® –

Univers Light
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@=¼½¾⅓⅔■°★†✓□_‡►"§☆¶●÷◄'©×±® –

Univers Light Italic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ ‡ ► " § ☆ ¶ ● ÷ ◄ ' © × ± ® –

Univers Extrabold Expanded
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ ‡ ► " § ☆ ¶ ● ÷ ◄ ' © × ± ® –

Univers Ultrabold Expanded
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}| + @ = ¼ ½ ¾ ⅓ ⅔ ■ ° ★ † ✓ □ ‡ ► " § ☆ ¶ ● ÷ ◄ ' © × ± ® –

Eras Medium
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789¢ ₵ □ __ /1234567890 ®

Eras Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789¢ ₵ □ __ /1234567890 ®

Eras Demibold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789¢ ₵ □ __ /1234567890 ®

Eras Ultrabold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789¢ ₵ □ __ /1234567890 ®

Eras Outline
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼ ½ ¾ ⅓ ⅔ $ ★ ✓ 0123456789¢ ₵ □ __ /1234567890 ®

Helios
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'*–•/&—$
%#ˆ˜+@=¡¸°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

Helios Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'*–•/&—$
%#ˆ˜+@=¡¸°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

Helios Semibold Outline
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'*–•/&—$
%#ˆ˜+@=¡¸°´ı†¨¿¢ˆ__˜ßÇ‡»ªç÷é«©×º±®–

University Roman
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@= ¼ ½ ¾ ⅓ ⅔ ■°★†✓□__‡►″§☆¶● ÷ ◄′©× ±® –

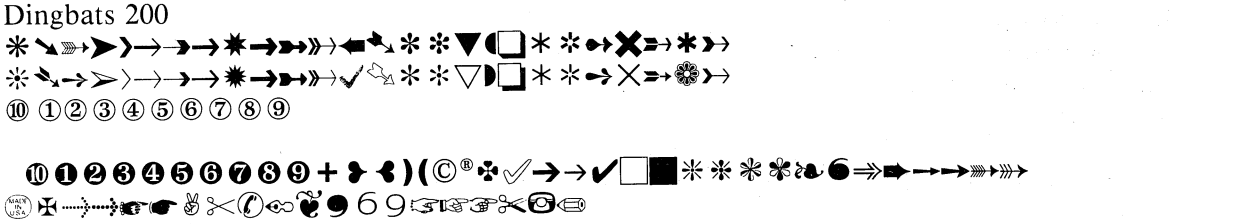
University Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@= ¼ ½ ¾ ⅓ ⅔ ■°★†✓□__‡►″§☆¶● ÷ ◄′©× ±® –

STENCIL
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼½¾⅓⅔$★✓0123456789ᶜ©□__/1234567890®

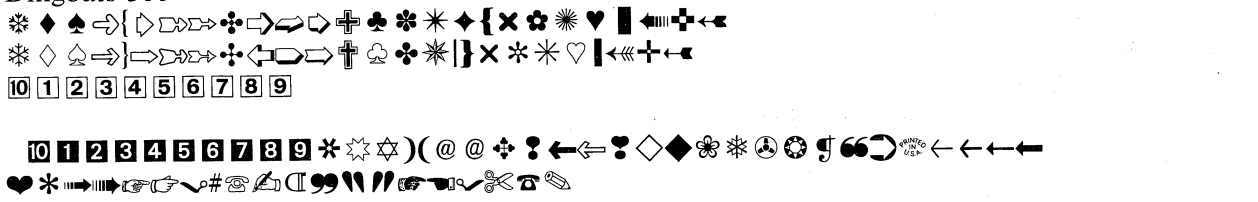
STENCIL OUTLINE
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
.,;:-(!)'?'[*]–•/&—$
% ¼½¾⅓⅔$★✓0123456789ᶜ©□__/1234567890®

Clearface Contour
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]–•/&—$
%# {}|+@= ¼½¾⅓⅔■°★†✓□__‡►″§☆¶● ÷ ◄′©× ±® –

Dingbats 200



Dingbats 300



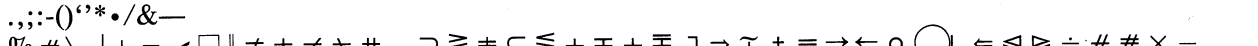
Script Text

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

.,;:-()‘’*•/&—



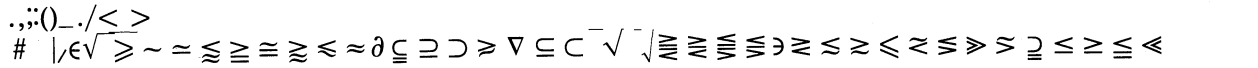
Full Size Greek Text

ABαΔΕΦΓΗΙφΚΛΜΝΟΠΩΡΣΤΥΘΨΧΞΖ

αβϑδεφγηισκλμνοπωρστυθψχξζ

0123456789

.,;:()_./< >



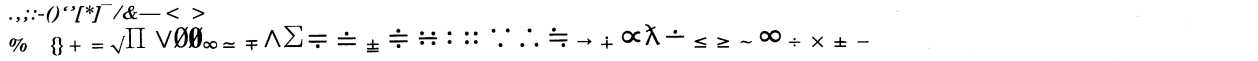
Inferior English Times Italic

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

.,;:-()‘’[*]¯/&— < >



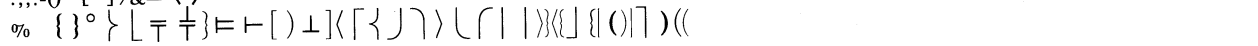
Inferior English Times

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

.,;:-0‘’[*]/&—〈 〉



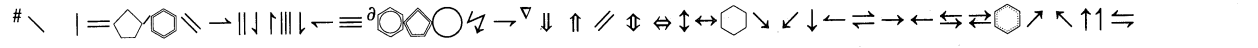
Superior Greek and Chemistry

ABαΔΕΦΓΗΙφΚΛΜΝΟΠΩΡΣΤΥΘΨΧΞΖ

αβϑδεφγηισκλμνοπωρστυθψχξζ

0123456789

.,;:0–· ⁄ —



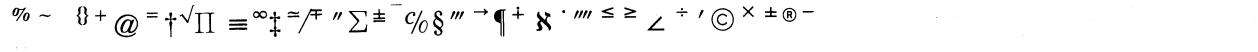
Superior English Times Italic

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

.,;:-()‘’[*]/&— < >



Inferior Greek

ΑΒαΔΕΦΓΗΙφΚΛ ΜΝΟΠΩΡΣΤΥΘΨΧΞΖ

αβϑδεϕγηισϰλμνοπωϱςτυθψχξζ

0123456789

.,;:0_./

\# |°__,∩☿ ♁ℓ♃ ♄ |-- Ψ ‖∂∪Å Æ|▿℞ ℄ ♂ ♀ ƒ ⌇ ℞ ≙≚≙ℋ | ‖/ ☋ ‖/ ∴ ℇ ∵ ⋮··· ¦ †

Superior English Times
ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

.,;:-0''[*]/&—$

% §£¢⊘⊙∰∮∯∮○∅ ≏ ≎ ∰∃∀∰∰∰⊗∮∮∰ ∰∫∱⊕∮∰ ∰∱

Phonetic English
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
.,;:-(!)'?'[*]-/&—$
˜ ˆ ˋ ˚ ˛ ı ˊ ¨ o ˘ ˜ ˉ ˗ ˇ , ˛ ˆ ȷ ˊ o͝o ˙ ˇ _ o͞o t͟h ˘ ˇ ̮

International Phonetic Alphabet
ɒɲɼɖəđɢɧɨɥʞɭɯɡ̣ɵɵʕɹʂʈðɲʍχ̢ƛʐ
ɑɳçɖɘɟɡħɨɦƙɬɱŋǫøɐɽʂʈʉʋsχʎʐ

.,-
˳̃\ ˆ ˋ ̩ ˚ ɪ ˊ ¨ ̰ ʃβɾ ˉ ɛçɬ ′′ ̃ʒ ˊ ɚʌɓɫ ɺɤɣ ' ˧ ˇ ʑʑ ʕ ƥ ɜɔɬ ʒɝʊ ′ ɠ ˫ ˔ ˕ ϱθʔ ˛ ɔæÆɭ ˘ ʁɸœ

APPENDIX D
Font-Group/Character Availability Table

As an alternative to Appendix C, the following cross-reference table is provided to check for the availability of frequently-used characters in different fonts. This is *not* an exhaustive list.

For the purposes of this Appendix, the fonts are broken into "font-groups", i.e. collections of fonts which contain the same sets of characters. The divisions are as follows:

Group A: Garth Graphic, Souvenir, Rockwell, Eras, Stencil
(All members of each family)

Group B: English Times, Grouch, Korinna Extrabold, Univers, University, Clearface Contour
(All members of each family)

Group C: Century, Helios
(All members of each family)

Group D: Baskerville, Baskerville Bold, Mallard, Mallard Bold

Group E: Baskerville Italic, Baskerville Bold Italic, Mallard Italic, Mallard Bold Italic

Group F: Serif Gothic
(All members of the family)

Group G: Tiffany
(All members of the family)

All other fonts have an individual collection of characters, and must be treated individually.

In the following table, a tick indicates that the character is available.

Character	*Group A*	*Group B*	*Group C*	*Group D*	*Group E*	*Group F*	*Group G*	*Flor'ne Script*	*Goudy Ex'bold*
Upper-case characters									
A	✓	✓	✓	✓	✓	✓	✓	✓	✓
B	✓	✓	✓	✓	✓	✓	✓	✓	✓
C	✓	✓	✓	✓	✓	✓	✓	✓	✓
D	✓	✓	✓	✓	✓	✓	✓	✓	✓
E	✓	✓	✓	✓	✓	✓	✓	✓	✓
F	✓	✓	✓	✓	✓	✓	✓	✓	✓
G	✓	✓	✓	✓	✓	✓	✓	✓	✓
I	✓	✓	✓	✓	✓	✓	✓	✓	✓
J	✓	✓	✓	✓	✓	✓	✓	✓	✓
K	✓	✓	✓	✓	✓	✓	✓	✓	✓
L	✓	✓	✓	✓	✓	✓	✓	✓	✓
M	✓	✓	✓	✓	✓	✓	✓	✓	✓
N	✓	✓	✓	✓	✓	✓	✓	✓	✓
O	✓	✓	✓	✓	✓	✓	✓	✓	✓
P	✓	✓	✓	✓	✓	✓	✓	✓	✓
Q	✓	✓	✓	✓	✓	✓	✓	✓	✓
R	✓	✓	✓	✓	✓	✓	✓	✓	✓
S	✓	✓	✓	✓	✓	✓	✓	✓	✓
T	✓	✓	✓	✓	✓	✓	✓	✓	✓
U	✓	✓	✓	✓	✓	✓	✓	✓	✓
V	✓	✓	✓	✓	✓	✓	✓	✓	✓
W	✓	✓	✓	✓	✓	✓	✓	✓	✓
X	✓	✓	✓	✓	✓	✓	✓	✓	✓
Y	✓	✓	✓	✓	✓	✓	✓	✓	✓
Z	✓	✓	✓	✓	✓	✓	✓	✓	✓
Lower-case characters									
a	✓	✓	✓	✓	✓	✓	✓	✓	✓
b	✓	✓	✓	✓	✓	✓	✓	✓	✓
c	✓	✓	✓	✓	✓	✓	✓	✓	✓
d	✓	✓	✓	✓	✓	✓	✓	✓	✓
e	✓	✓	✓	✓	✓	✓	✓	✓	✓
f	✓	✓	✓	✓	✓	✓	✓	✓	✓
g	✓	✓	✓	✓	✓	✓	✓	✓	✓
h	✓	✓	✓	✓	✓	✓	✓	✓	✓
i	✓	✓	✓	✓	✓	✓	✓	✓	✓
j	✓	✓	✓	✓	✓	✓	✓	✓	✓
k	✓	✓	✓	✓	✓	✓	✓	✓	✓
l	✓	✓	✓	✓	✓	✓	✓	✓	✓
m	✓	✓	✓	✓	✓	✓	✓	✓	✓
n	✓	✓	✓	✓	✓	✓	✓	✓	✓
o	✓	✓	✓	✓	✓	✓	✓	✓	✓
p	✓	✓	✓	✓	✓	✓	✓	✓	✓
q	✓	✓	✓	✓	✓	✓	✓	✓	✓
r	✓	✓	✓	✓	✓	✓	✓	✓	✓
s	✓	✓	✓	✓	✓	✓	✓	✓	✓
t	✓	✓	✓	✓	✓	✓	✓	✓	✓
u	✓	✓	✓	✓	✓	✓	✓	✓	✓
v	✓	✓	✓	✓	✓	✓	✓	✓	✓
w	✓	✓	✓	✓	✓	✓	✓	✓	✓
x	✓	✓	✓	✓	✓	✓	✓	✓	✓
y	✓	✓	✓	✓	✓	✓	✓	✓	✓
z	✓	✓	✓	✓	✓	✓	✓	✓	✓

Character	*Group A*	*Group B*	*Group C*	*Group D*	*Group E*	*Group F*	*Group G*	*Flor'ne Script*	*Goudy Ex'bold*
Numerics									
0	✓	✓	✓	✓	✓	✓	✓	✓	✓
1	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fractions									
1/8				✓					
1/4	✓	✓		✓				✓	
3/8				✓					
1/2	✓	✓		✓				✓	
5/8				✓					
3/4	✓	✓		✓				✓	
7/8				✓					
1/3	✓	✓		✓				✓	
2/3	✓	✓		✓				✓	
Superior characters									
0	✓					✓	✓		✓
1	✓					✓	✓		✓
2	✓					✓	✓		✓
3	✓					✓	✓		✓
4	✓					✓	✓		✓
5	✓					✓	✓		✓
6	✓					✓	✓		✓
7	✓					✓	✓		✓
8	✓					✓	✓		✓
9	✓					✓	✓		✓
$	✓					✓	✓		✓
¢	✓					✓	✓		✓
a			✓						
o			✓						
Inferior characters									
1	✓								
2	✓								
3	✓								
4	✓								
5	✓								
6	✓								
7	✓								
8	✓								
9	✓								
0	✓								

Character	*Group A*	*Group B*	*Group C*	*Group D*	*Group E*	*Group F*	*Group G*	*Flor'ne Script*	*Goudy Ex'bold*
European accents									
ç			✓				✓		
Ç			✓				✓		
ˆ				✓	✓				
ˆ			✓	✓	✓				
ˆ			✓	✓	✓		✓		
¨			✓	✓	✓				
¨			✓	✓	✓		✓		
`			✓	✓	✓				
`			✓	✓	✓		✓		
~			✓	✓	✓				
~			✓	✓	✓		✓		
é			✓						
i			✓	✓	✓		✓		
c			✓	✓	✓				
j			✓	✓	✓				
ß			✓	✓	✓				
Punctuation and other symbols									
.	✓	✓	✓	✓	✓	✓	✓	✓	✓
,	✓	✓	✓	✓	✓	✓	✓	✓	✓
;	✓	✓	✓	✓	✓	✓	✓	✓	✓
:	✓	✓	✓	✓	✓	✓	✓	✓	✓
?	✓	✓	✓	✓	✓	✓	✓	✓	✓
!	✓	✓	✓	✓	✓	✓	✓	✓	✓
(	✓	✓	✓	✓	✓	✓	✓	✓	✓
)	✓	✓	✓	✓	✓	✓	✓	✓	✓
*	✓	✓	✓	✓	✓	✓	✓	✓	✓
&	✓	✓	✓	✓	✓	✓	✓	✓	✓
-	✓	✓	✓	✓	✓	✓	✓	✓	✓
(Hyphen)									
/	✓	✓	✓	✓	✓	✓	✓	✓	✓
`	✓	✓	✓	✓	✓	✓	✓	✓	✓
`	✓	✓	✓	✓	✓	✓	✓	✓	✓
$	✓	✓	✓	✓	✓	✓	✓	✓	✓
|	✓	✓		✓	✓	✓	✓	✓	✓
|	✓	✓		✓	✓	✓	✓	✓	✓
—	✓	✓	✓	✓	✓	✓	✓	✓	✓
(Em-dash)									
—	✓	✓	✓	✓	✓	✓	✓	✓	✓
(Base-line rule)									
–	✓	✓	✓		✓				
(En-dash)									
+		✓	✓		✓				
–		✓	✓		✓				
(Minus sign)									
×		✓	✓		✓				
(Multiplication sign)									
÷		✓	✓		✓				
=		✓	✓		✓				
@		✓	✓						
%	✓	✓	✓	✓	✓	✓	✓		✓
#		✓	✓	✓				✓	
•	✓	✓	✓	✓					

APPENDIX E
Access Codes — Font Blueprints

As in Appendix D, the available fonts have been placed into "font-groups', where any face in any particular "group" has the same characters, each of which is characterised by a unique access code.

Further, in this appendix, it is necessary to present the display in two parts:

In Part A, the more common "text" styles (Groups A-E) are displayed, and for all of the faces in these groups it is true to say that a particular access code will either produce a consistent character or nothing, i.e., for example, the code + [will either produce [(if this character exists in the current font) or else it will be ignored (if this character does not exist in the current font) ... it *cannot* produce a different character if the current face is any of the faces from any of the groups, Group A-E.

In Part B, all the other faces are treated, and for these it is not possible to maintain complete consistency, neither with access codes used in Part A nor with access codes within Part B — each face has to be taken individually.

The divisions are as follows:

PART A

Group A: Garth Graphic, Souvenir, Rockwell, Eras, Stencil
(All members of each family)

Group B: English Times, Grouch, Korinna Extrabold, Univers, University, Clearface Contour
(All members of each family)

Group C: Century, Helios
(All members of each family)

Group D: Baskerville, Baskerville Bold, Mallard, Mallard Bold

Group E: Baskerville Italic, Baskerville Bold Italic, Mallard Italic, Mallard Bold Italic

PART B

Serif Gothic, Tiffany, Florentine Script, Goudy Extrabold, Dingbats 200, Dingbats 300, Mathematics/Greek, Phonetics (all members of these families).

In the following tables, a character in a column indicates the character associated with the nominated access code in the left-hand column; the absence of a character denotes that there is no character in that font corresponding to the particular access code.

In the table for Part A, the "representative" member of each group is as follows:

Group A: Garth Graphic
Group B: English Times
Group C: Century Schoolbook
Group D: Baskerville
Group E: Baskerville Italic

PART A:

Access Code	*Group A*	*Group B*	*Group C*	*Group D*	*Group E*
A	A	A	A	A	*A*
B	B	B	B	B	*B*
C	C	C	C	C	*C*
D	D	D	D	D	*D*
E	E	E	E	E	*E*
F	F	F	F	F	*F*
G	G	G	G	G	*G*
H	H	H	H	H	*H*
I	I	I	I	I	*I*
J	J	J	J	J	*J*
K	K	K	K	K	*K*
L	L	L	L	L	*L*
M	M	M	M	M	*M*
N	N	N	N	N	*N*
O	O	O	O	O	*O*
P	P	P	P	P	*P*
Q	Q	Q	Q	Q	*Q*
R	R	R	R	R	*R*
S	S	S	S	S	*S*
T	T	T	T	T	*T*
U	U	U	U	U	*U*
V	V	V	V	V	*V*
W	W	W	W	W	*W*
X	X	X	X	X	*X*
Y	Y	Y	Y	Y	*Y*
Z	Z	Z	Z	Z	*Z*
a	a	a	a	a	*a*
b	b	b	b	b	*b*
c	c	c	c	c	*c*
d	d	d	d	d	*d*
e	e	e	e	e	*e*
f	f	f	f	f	*f*
g	g	g	g	g	*g*
h	h	h	h	h	*h*
i	i	i	i	i	*i*
j	j	j	j	j	*j*
k	k	k	k	k	*k*
l	l	l	l	l	*l*
m	m	m	m	m	*m*
n	n	n	n	n	*n*
o	o	o	o	o	*o*
p	p	p	p	p	*p*
q	q	q	q	q	*q*
r	r	r	r	r	*r*
s	s	s	s	s	*s*
t	t	t	t	t	*t*
u	u	u	u	u	*u*
v	v	v	v	v	*v*
w	w	w	w	w	*w*
x	x	x	x	x	*x*
y	y	y	y	y	*y*
z	z	z	z	z	*z*

Access Code	*Group A*	*Group B*	*Group C*	*Group D*	*Group E*
0	0	0	0	0	*0*
1	1	1	1	1	*1*
2	2	2	2	2	*2*
3	3	3	3	3	*3*
4	4	4	4	4	*4*
5	5	5	5	5	*5*
6	6	6	6	6	*6*
7	7	7	7	7	*7*
8	8	8	8	8	*8*
9	9	9	9	9	*9*
.	.	.	.	.	.
(Keyboard character period produces period)					
,	,	,	,	,	,
(Keyboard character comma produces comma)					
;	;	;	;	;	;
:	:	:	:	:	:
!	!	!	!	!	!
?	?	?	?	?	?
(	(	(	(	(	(
)	)	)	)	)	)
-	-	-	-	-	-
(Keyboard character hyphen produces hyphen)					
&	&	&	&	&	&
`	'	‘	‘	‘	‘
(Keyboard character back-quote produces single opening quote)					
'	'	'	'	'	'
(Keyboard character apostrophe produces closing quote/apostrophe)					
$	$	$	$	$	$
%	%	%	%	%	%
{		{			
}		}			
—	—	—	—	—	—
(Keyboard character underscore produces em-dash)					
*	*	*	*	*	*
#		#	#	#	
"			`	`	`
(Keyboard character double-quote produces lower-case grave accent)					
^			^	^	^
(Keyboard character up-arrow produces lower-case circumflex)					
|		|			
(Keyboard character vertical bar produces vertical rule)					
~			~	~	~
(Keyboard character tilde produces lower-case tilde)					
//	/	/	/	/	/
(Oblique character is produced)					
/[	[	[		[	[
/]	]	]		]	]
/@		@	@		
/ +		+	+		+
/ =		=	=		=
/1				⅛	
/2	¼	¼		¼	
/3				⅜	
/4	½	½		½	
/5				⅝	

Access Code	Group A	Group B	Group C	Group D	Group E
/6	¾	¾		¾	
/7				⅞	
/8	⅓	⅓		⅓	
/9	⅔	⅔		⅔	
+ !			¡	¡	¡
(Inverted exclamation mark is produced)					
+ "			`	`	`
(Plus-double quote combination produces upper-case grave accent)					
+ #		■			■
+ $	$				
(Superior $-symbol is produced)					
+ <	¢				
(Keyboard combination plus-left angle bracket produces superior cent-symbol)					
+ %		°	°		°
(Produces degree-symbol)					
+ '			´	´	´
(Keyboard combination plus-apostrophe produces upper-case acute accent)					
+ (			ı	ı	ı
(Produces dotless-i)					
+)				¸	¸
(Produces cedilla accent)					
+ *	★	★			
+ &				£	£
+ +		†	†		†
+ ,			´	´	´
(Keyboard combination plus-comma produces lower-case acute accent)					
+ -	–	–	–		–
(Keyboard combination plus-hyphen produces en-dash)					
+ 0	0				
(Superior 0 produced ... similarly for other superior numerals below)					
+ 1	1				
+ 2	2				
+ 3	3				
+ 4	4				
+ 5	5				
+ 6	6				
+ 7	7				
+ 8	8				
+ 9	9				
+ .	•	•	•	•	
(Produces en-bullet)					
+ /	✓	✓			
+ :			¨	¨	¨
(Produces upper-case umlaut)					
+ ;			¨	¨	¨
(Produces lower-case umlaut)					
+ ?			¿	¿	¿
(Produces inverted question-mark)					
+ @	¢		¢	¢	
+ [	□	□		□	
+ ^			ˆ	ˆ	ˆ
(Keyboard combination plus-up arrow produces upper-case circumflex)					
+ _	_	_	_	_	_
(Keyboard combination plus-underscore produces base-line rule)					
+ ~			˜	˜	˜
(Keyboard combination plus-tilde produces upper-case tilde)					

Access Code	*Group A*	*Group B*	*Group C*	*Group D*	*Group E*
+ A				1	*1*
(Produces fit-1 character)					
+ B			ß	ß	*ß*
+ C			Ç		
+ D		‡	‡		
+ E					·
(Produces centred-dot character)					
+ F	/				
(Produces fraction-bar character)					
+ G		►			
+ H		″			
(Produces seconds-symbol)					
+ I				ffi	*ffi*
+ J			»		
+ L				ffl	*ffl*
+ M		§			§
+ O		☆			
+ P		¶			
+ Q	$_1$				
(Produces inferior numeral 1 ... similarly for others below)					
+ R	$_2$				
+ S	$_3$				
+ T	$_4$				
+ U	$_5$				
+ V	$_6$				
+ W	$_7$				
+ X	$_8$				
+ Y	$_9$				
+ Z	$_0$				
+ a			^{a}		
(Produces superior character a)					
+ b		●			
+ c			ç		
+ d		÷	÷		÷
+ e			é		
+ f				ff	*ff*
+ g		◄			
+ h		′			′
(Produces minutes-symbol)					
+ i				fi	*fi*
+ j			«		
+ k		©	©		©
+ l				fl	*fl*
+ m		×	×		×
(Produces multiplication-symbol)					
+ o			^{o}		
(Produces superior character o)					
+ p		±	±		
+ r	®	®	®		®
+ s		−	−		−
(Produces minus-symbol)					

The following keyboard characters and combinations produce *no* character:
< > + > + K + N + n + q + s + t + u + v + w + x + y + z

PART B:

Section (i)

Access Code	*Serif Gothic*	*Tiffany*	*Flor'ine Script*	*Goudy E'bold*	*Dingbats 200*	*Dingbats 300*	*English Phon.*	*Inter. Phon.*
A	A	A	𝒜	**A**	✻	❄	A	ɒ
B	B	B	𝐵	**B**	➘	♦	B	ɳ
C	C	C	𝒞	**C**	➳	♠	C	ɾ
D	D	D	𝒟	**D**	➤	⇨	D	ɖ
E	E	E	ℰ	**E**	❯	❴	E	ə
F	F	F	ℱ	**F**	→	⇨	F	đ
G	G	G	𝒢	**G**	➔	⇨	G	ɢ
H	H	H	ℋ	**H**	→	⇨	H	ɧ
I	I	I	ℐ	**I**	✷	✣	I	ɨ
J	J	J	𝒥	**J**	➔	⇨	J	ɥ
K	K	K	𝒦	**K**	➙	▰	K	ʞ
L	L	L	ℒ	**L**	➳	⇨	L	ɭ
M	M	M	ℳ	**M**	⬅	✙	M	ɯ
N	N	N	𝒩	**N**	➴	♣	N	ɡ
O	O	O	𝒪	**O**	✻	✽	O	ʊ
P	P	P	𝒫	**P**	✼	✳	P	ɵ
Q	Q	Q	𝒬	**Q**	▼	✦	Q	ɋ
R	R	R	ℛ	**R**	◖	❴	R	ɹ
S	S	S	𝒮	**S**	❑	✖	S	ʂ
T	T	T	𝒯	**T**	✳	✿	T	ʈ
U	U	U	𝒰	**U**	✲	✺	U	ð
V	V	V	𝒱	**V**	➙	♥	V	ɲ
W	W	W	𝒲	**W**	✖	▮	W	ʍ
X	X	X	𝒳	**X**	➳	⬅	X	χ
Y	Y	Y	𝒴	**Y**	✱	✣	Y	ƛ
Z	Z	Z	𝒵	**Z**	➙	⬅	Z	ʐ
a	a	a	𝒶	**a**	✼	❄	a	ɑ
b	b	b	𝒷	**b**	➴	♢	b	ɳ
c	c	c	𝒸	**c**	➔	♤	c	ç
d	d	d	𝒹	**d**	➢	⇨	d	ɖ
e	e	e	ℯ	**e**	❯	❵	e	ɐ
f	f	f	𝒻	**f**	→	⇨	f	ɟ
g	g	g	ℊ	**g**	→	⇨	g	g
h	h	h	𝒽	**h**	→	⇨	h	ħ
i	i	i	𝒾	**i**	✷	✣	i	ɨ
j	j	j	𝒿	**j**	➔	⇦	j	ɦ
k	k	k	𝓀	**k**	➙	▭	k	ʞ
l	l	l	𝓁	**l**	➳	⇨	l	ɬ
m	m	m	𝓂	**m**	✓	✝	m	ɱ
n	n	n	𝓃	**n**	➷	♧	n	ŋ
o	o	o	ℴ	**o**	✻	✤	o	ɷ
p	p	p	𝓅	**p**	✼	✳	p	ø
q	q	q	𝓆	**q**	▽	❘	q	ɐ
r	r	r	𝓇	**r**	◗	❵	r	ɾ
s	s	s	𝓈	**s**	❑	✖	s	ʂ
t	t	t	𝓉	**t**	✳	✲	t	ʈ
u	u	u	𝓊	**u**	✲	✳	u	ʉ
v	v	v	𝓋	**v**	➙	♡	v	ʊ
w	w	w	𝓌	**w**	✕	▮	w	ʂ
x	x	x	𝓍	**x**	➳	⬅	x	χ
y	y	y	𝓎	**y**	❁	✢	y	ʎ
z	z	z	𝓏	**z**	➙	⬅	z	ʐ

Access Code	*Serif Gothic*	*Tiffany*	*Flor'ine Script*	*Goudy E'bold*	*Dingbats 200*	*Dingbats 300*	*English Phon.*	*Inter. Phon.*
0	0	0	0	0	⑩	10	0	
1	1	1	1	1	①	1	1	
2	2	2	2	2	②	2	2	
3	3	3	3	3	③	3	3	
4	4	4	4	4	④	4	4	
5	5	5	5	5	⑤	5	5	
6	6	6	6	6	⑥	6	6	
7	7	7	7	7	⑦	7	7	
8	8	8	8	8	⑧	8	8	
9	9	9	9	9	⑨	9	9	
.	.	.	.	.			.	.
(Keyboard character period produces period)								
,	,	,	,	,			,	,
(Keyboard character comma produces comma)								
;	;	;	;	;			;	
:	:	:	:	:			:	
!	!	!	!	!			!	
?	?	?	?	?			?	
(	(	(	(	(			(	
)	)	)	)	)			)	
-	-	-	-	-			-	-
(Keyboard character hyphen produces hyphen)								
&	&	&	&	&			&	
‘	‘	‘	‘	‘			‘	
(Keyboard character back-quote produces single opening quote)								
’	’	’	’	’			’	
(Keyboard character apostrophe produces closing quote/apostrophe)								
$	$	$	$	$			$	
%	%	%		%			°	°
—	—	—	–	—			—	
(Keyboard character underscore produces em-dash)								
*	*	*	*	*			*	
#			#					
”			”	”			”	”
(Keyboard character double-quote)								
^			^				^	^
(Keyboard character up-arrow)								
¿							¦	¦
(Keyboard character vertical bar)								
~			~				~	~
(Keyboard character tilde)								
//	/	/	/	/			/	
(Oblique)								
/[	[	[	/	[			[	
/]	]	]	/	]			]	
/								
/ +								
/ =								
/1								
/2			½					
/3								
/4			½					
/5								

Access Code	*Serif Gothic*	*Tiffany*	*Flor'ine Script*	*Goudy E'bold*	*Dingbats 200*	*Dingbats 300*	*English Phon.*	*Inter. Phon.*
/6			¾					
/7								
/8			⅓					
/9								
+!								
+"								
+#								
+$	$	$		$				
+%			°				°	°
+'							'	
+(			'				1	1
+)								
+*								
+&		&	&					
+ +	†		+					
+,			,				,	,
+-			–				–	
+0	0	0		0	⓾	10		
+1	1	1	1	1	❶	1		
+2	2	2	2	2	❷	2		
+3	3	3	3	3	❸	3		
+4	4	4	4	4	❹	4		
+5	5	5		5	❺	5		
+6	6	6		6	❻	6		
+7	7	7		7	❼	7		
+8	8	8		8	❽	8		
+9	9	9		9	❾	9		
+.			.					
+/								
+:								
+;			..				..	..
+ =								
+?								
+@	¢	¢	%	¢				
+[								
+]								
+^							^	^
+—	—	—	—	—				
+‘								
+							'	
+<	¢	¢		¢			(	
+>							)	
+~							~	~
\								\

Access Code	*Serif Gothic*	*Tiffany*	*Flor'ine Script*	*Goudy E'bold*	*Dingbats 200*	*Dingbats 300*	*English Phon.*	*Inter. Phon.*
+ A								
+ B								
+ C								
+ D								
+ E								
+ F								
+ G								
+ H								
+ I								
+ J								
+ K								
+ L								
+ M								
+ N								
+ O								
+ P								
+ Q								
+ R								
+ S								
+ T								
+ U								
+ V								
+ W								
+ X								
+ Y								
+ Z								
+ a								
+ b								
+ c								
+ d								
+ e								
+ f								
+ g								
+ h								
+ i								
+ j								
+ k								
+ l								
+ m								
+ n								
+ o								
+ p								
+ q								
+ r								
+ s								
+ t								
+ u								
+ v								
+ w								
+ x								
+ y								
+ z								

PART B:

Section (ii)

Access Code	*Font 900*	*Font 910*	*Font 920*	*Font 930*	*Font 940*	*Font 950*	*Font 960*	*Font 970*
A	𝔄	Α	*Α*	Α	Α	*Α*	Α	Α
B	𝔅	Β	*Β*	Β	Β	*Β*	Β	Β
C	ℭ	α	*C*	C	α	*C*	α	C
D	𝔇	Δ	*D*	D	Δ	*D*	Δ	D
E	𝔈	Ε	*Ε*	Ε	Ε	*Ε*	Ε	Ε
F	𝔉	Φ	*F*	F	Φ	*F*	Φ	F
G	𝔊	Γ	*G*	G	Γ	*G*	Γ	G
H	ℌ	Η	*Η*	Η	Η	*Η*	Η	Η
I	ℑ	Ι	*Ι*	Ι	Ι	*Ι*	Ι	Ι
J	𝔍	φ	*J*	J	φ	*J*	φ	J
K	𝔎	Κ	*Κ*	Κ	Κ	*Κ*	Κ	Κ
L	ℒ	Λ	*L*	L	Λ	*L*	Λ	L
M	ℳ	Μ	*Μ*	Μ	Μ	*Μ*	Μ	Μ
N	𝔑	Ν	*Ν*	Ν	Ν	*Ν*	Ν	Ν
O	𝔒	Ο	*Ο*	Ο	Ο	*Ο*	Ο	Ο
P	𝔓	Π	*Π*	Π	Π	*Π*	Π	Π
Q	𝔔	Ω	*Q*	Q	Ω	*Q*	Ω	Q
R	ℜ	Ρ	*Ρ*	Ρ	Ρ	*Ρ*	Ρ	Ρ
S	𝔖	Σ	*S*	S	Σ	*S*	Σ	S
T	𝔗	Τ	*Τ*	Τ	Τ	*Τ*	Τ	Τ
U	𝔘	ϒ	*U*	U	ϒ	*U*	ϒ	U
V	𝔙	Θ	*V*	V	Θ	*V*	Θ	V
W	𝔚	Ψ	*W*	W	Ψ	*W*	Ψ	W
X	𝔛	Χ	*Χ*	Χ	Χ	*Χ*	Χ	Χ
Y	𝔜	Ξ	*Υ*	Υ	Ξ	*Υ*	Ξ	Υ
Z	ℨ	Ζ	*Ζ*	Ζ	Ζ	*Ζ*	Ζ	Ζ
a	𝔞	α	*α*	a	α	*a*	α	a
b	𝔟	β	*b*	b	β	*b*	β	b
c	c	ϑ	*c*	c	ϑ	*c*	ϑ	c
d	𝔡	δ	*d*	d	δ	*d*	δ	d
e	*e*	ϵ	*e*	e	ϵ	*e*	ϵ	e
f	𝔣	ϕ	*f*	f	ϕ	*f*	ϕ	f
g	𝔤	γ	*g*	g	γ	*g*	γ	g
h	𝔥	η	*h*	h	η	*h*	η	h
i	*i*	ι	*i*	i	ι	*i*	ι	i
j	*j*	σ	*j*	j	σ	*j*	σ	j
k	𝔨	χ	*k*	k	χ	*k*	χ	k
l	ℓ	λ	*l*	l	λ	*l*	λ	l
m	*m*	μ	*m*	m	μ	*m*	μ	m
n	*n*	ν	*n*	n	ν	*n*	ν	n
o	*o*	o	*o*	o	o	*o*	o	o
p	𝔭	π	*p*	p	π	*p*	π	p
q	𝔮	ω	*q*	q	ω	*q*	ω	q
r	*r*	ϱ	*r*	r	ϱ	*r*	ϱ	r
s	𝔰	ς	*s*	s	ς	*s*	ς	s
t	*t*	τ	*t*	t	τ	*t*	τ	t
u	*u*	υ	*u*	u	υ	*u*	υ	u
v	*v*	θ	*v*	v	θ	*v*	θ	v
w	*w*	ψ	*w*	w	ψ	*w*	ψ	w
x	*x*	χ	*x*	x	χ	*x*	χ	x
y	𝔶	ξ	*y*	y	ξ	*y*	ξ	y
z	*z*	ζ	*z*	z	ζ	*z*	ζ	z

Access Code	*Font 900*	*Font 910*	*Font 920*	*Font 930*	*Font 940*	*Font 950*	*Font 960*	*Font 970*
0	0	0	*0*	0	0	*0*	0	0
1	1	1	*1*	1	1	*1*	1	1
2	2	2	*2*	2	2	*2*	2	2
3	3	3	*3*	3	3	*3*	3	3
4	4	4	*4*	4	4	*4*	4	4
5	5	5	*5*	5	5	*5*	5	5
6	6	6	*6*	6	6	*6*	6	6
7	7	7	*7*	7	7	*7*	7	7
8	8	8	*8*	8	8	*8*	8	8
9	9	9	*9*	9	9	*9*	9	9
.	.	.	.	.	.	.	.	.
(Keyboard character period produces period)								
,	,	,	,	,	,	,	,	,
(Keyboard character comma produces comma)								
;	;	;	;	;	;	;	;	;
:	:	:	:	:	:	:	:	:
!								
?								
(	(	(	(	(	(	(	(	(
)	)	)	)	)	)	)	)	)
-	-		-	-		-		-
(Keyboard character hyphen produces hyphen)								
&	&		&	&		&		&
‘	‘		‘	‘		‘		‘
(Keyboard character back-quote produces single opening quote)								
’	’		’	’		’		’
(Keyboard character apostrophe produces closing quote/apostrophe)								
$								$
%	%		%	%		%		%
			{	{		{		{
			}	}		}		}
—	—		—	—	—	—		—
(Keyboard character underscore produces em-dash)								
*	*		*	*		*		*
#	#	#			#		#	
”								
(Keyboard character double-quote)								
^								
(Keyboard character up-arrow)								
|	|	|			|		|	
(Keyboard character vertical bar produces vertical rule)								
~						~		
(Keyboard character tilde)								
//	/	/	/	/	/	/	/	/
(Oblique)								
/[			[	[		[		[
/]			]	]		]		]
/						@		
/ +	+		+			+		
/ =	=		=		=	=		

Access Code	*Font 900*	*Font 910*	*Font 920*	*Font 930*	*Font 940*	*Font 950*	*Font 960*	*Font 970*
/>		>	>	⟩		>		
/<		<	<	⟨		<		
+ %				°			°	
+ &								£
+ +						†		
+ -		–	–		–		–	
+ .	•	·			·		·	
+ /	✓		√			√		
+ ⅝								¢
+ [	□							
+ —							—	
\	\				\			

Access Code	*Font 900*	*Font 910*	*Font 920*	*Font 930*	*Font 940*	*Font 950*	*Font 960*	*Font 970*
+*A*	∥	⁄	Π	}	⬠	Π	′	⊘
+*B*	≠	∈	∨	⌊	′	≡	∩	⊙
+*C*	±	√	∅	⌈	⬡	∞	♀	⨑
+*D*	≠		∅	⌈	⫽	‡	⊙	∫
+*E*	∓	⩾	∞	}	→	≃	ℓ	∮
+*F*	#	~	≃	⊨	‖	⁄	2l	⁄
+*G*	⌐	≃	∓	⊢	↓	∓	ħ	∮
+*H*	⊇	⩾	∧	[	↑	″	|	∮
+*I*	⩾	⩾	Σ	)	⫴	Σ	--	○
+*J*	⩾	⩾	∓	⊥	↓	±	Ψ	∅
+*K*	⊆	⩾	≐	]	←		‖	≐
+*L*	⩽	⩽	±	〈	≡	%	⁄	≎
+*M*	±	≈	∓	⌈	∂	§	∂	⨑
+*N*	∓	∂	∺	}	⬡	‴	∪	∃
+*O*	±	⊆	:	⌋	⬠	→	Å	∀
+*P*	∓	⊇	::	⌉	◯	¶	Æ	⫴
+*Q*	⌝	⊃	∵	〉	↯	+	|	⫴
+*R*	⇒	⩾	∴	⌊	→	ℵ	▿	|
+*S*	∓	∇	≒	⌈	▿	·	R	⫴
+*T*	±	⊆	→	|	⇓	′′′′	₺	⊗
+*U*	≡	⊂	+	|	⇑	≤	♂	∮
+*V*	⇄		∝	〉	⫽	≥	♀	∮
+*W*	⇆	√	ϰ	}	⇳	∠	f	⨌
+*X*	○		÷	〈	⇔		≀	⨑
+*Y*	◯	√	≤	{	↕		P	|
+*Z*	⌊	⩾	≥	⌊	↔		≙	∫
+*a*	⇐	⩽	~	{	⬡		≚	∮
+*b*	⩽	⩽	∞	|	↘		≙	|
+*c*	⩾	⩾		(	↙		ℑ	|
+*d*	÷	⩾	÷	)	↓	÷	|	⊕
+*e*	#	⩾		|	←		‖	∮
+*f*	⋕	⩽		⌉	⇄		/	⨌
+*g*		⩾		)	→		⨯	⨌
+*h*		⩽		(	←	′	|	∗
+*i*		⩾		〈	⇆		/	|
+*j*		⩽			⇄		⋰	
+*k*		⩾			⬡	©		
+*l*		⩽			↗			
+*m*	×	⊇	×		↖	×	ε	
+*n*		⩽			↑		⋱	
+*o*		⩾			↿		⋮	
+*p*		⩽	±		⇋	±	....	
+*q*		⩽					|	
+*r*						®	†	
+*s*	–		–			–		

APPENDIX F
Sample of Pointsizes

The following chart may assist in determining or deciding on appropriate point sizes in some selected faces.

Pointsize	*English Times*	*Univers Medium*	*Baskerville*	*Souvenir Medium*
5	x	x	x	x
	4	4	4	4
	X	X	X	X
6	x	x	x	x
	4	4	4	4
	X	X	X	X
7	x	x	x	x
	4	4	4	4
	X	X	X	X
8	x	x	x	x
	4	4	4	4
	X	X	X	X
9	x	x	x	x
	4	4	4	4
	X	X	X	X
10	x	x	x	x
	4	4	4	4
	X	X	X	X
11	x	x	x	x
	4	4	4	4
	X	X	X	X
12	x	x	x	x
	4	4	4	4
	X	X	X	X
14	x	x	x	x
	4	4	4	4
	X	X	X	X
16	x	x	x	x
	4	4	4	4
	X	X	X	X
18	x	x	x	x
	4	4	4	4
	X	X	X	X

Pointsize	*English Times*	*Univers Medium*	*Baskerville*	*Souvenir Medium*
20	x	x	x	x
	4	4	4	4
	X	X	X	X
24	x	x	x	x
	4	4	4	4
	X	X	X	X
30	x	x	x	x
	4	4	4	4
	X	X	X	X
36	x	x	x	x
	4	4	4	4
	X	X	X	X
42	x	x	x	x
	4	4	4	4
	X	X	X	X
54	x	x	x	x
	4	4	4	4
	X	X	X	X