



# **Guide to Quikscript**

**T. Graham Freeman**

October 1998

Technical Report CS01/98

School of Computer Science  
Australian Defence Force Academy

[g-freeman@adfa.edu.au](mailto:g-freeman@adfa.edu.au)

This document supersedes and expands on Technical Reports CS91/5 and CS05/94.  
Most recent revision January 2007.

## Guide to Quikscript

Copyright © 2007 by Graham Freeman

In past releases, Quikscript and this Guide were marked as not to be used for commercial purposes without the written consent of the author. With the October 2005 release of Quikscript, this is relaxed so that now both Quikscript and this document may be freely used or transmitted provided that this authorship and copyright notice is retained unmodified. The author accepts no legal liability for any damage or loss caused by the use of Quikscript or any software or documentation related to it.

Quikscript has been developed progressively since 1985, when it was used prior to the widespread availability of word-processing software to format documents and produce overhead transparencies, relying only on the existence of PostScript printers to interpret the embedded layout marks that determine the appearance of the final pages.

## Introduction

Traditionally, the choice when creating a document has been to use a word processor on a personal computer, or to use a typesetting package on a more powerful machine. Quikscript is a typesetting package that does not run on a mainframe computer, but rather within the printer itself. The advantage of this is portability. Any computer can be used for document preparation, provided that a PostScript printer is available.

The second advantage that Quikscript provides is precision of control over output. Because it is written in PostScript, it is interpreted at run-time within the printer. It is possible to create documents that modify the Quikscript program during execution. It is very easy to include other PostScript programs or fragments with Quikscript. It is possible to use special PostScript fonts, such as hand-generated ones. Graphics generated from a variety of sources can be easily included, as can text output from computer programs.

Document preparation consists of typing the document into an editor or word processor on your computer. Then layout instructions are added to the document. Finally the Quikscript program is sent to the PostScript printer together with the document file. Quikscript executes in the printer, reading your document, and acting on any layout instructions. It breaks the text into words, and fits them into lines of output across the page. If the computer has a PostScript previewer such as *ghostscript* or *gsview*, the process of creating a document can be assisted by running the previewer on Quikscript and your document. This can more quickly display errors in layout instructions. *ghostscript* also enables non-PostScript printers to be used to print Quikscript documents.

This introductory guide is designed to show how to use the basic features of Quikscript with simple examples. It proceeds to show more sophisticated layouts and how they might be achieved. Finally it covers examples of how PostScript programs can be embedded in your document to achieve special effects. This is a major strength of Quikscript. The commands of Quikscript are formally described in Appendix A.

## Commands for Simple Layout

### Paragraphs, lines

Under normal use, Quikscript takes input lines of text, breaks them up into words, and tries to fill output lines with the words. To control this process, certain layout instructions can be used with the text. The instructions are separated from the text by being enclosed by percent signs, eg. %P% indicates a new paragraph; %L% indicates a new line; %NP% forces a new page.

For example, if we want to set text in this way:

```
PostScript is a programming language in which numbers or parameters
precede the operator that acts on them. Thus to add two numbers, 1.3 and
5.5, the instruction is
1.3 5.5 add
```

The numbers are placed on a stack, and the "add" operator acts on and removes the numbers from the stack, returning the result there.

It is necessary sometimes to place items on the stack well before an operator or procedure acts on them. It can then be most unclear to the reader just what an operator is doing. This causes many PostScript programs to be hard to read, and makes debugging the program difficult.

As a general comment, 90% of programmers have not worked with a language like PostScript before.

we could type it thus:

```
PostScript is a programming
language in which numbers or parameters
precede the operator that acts on them.
Thus to add two numbers, 1.3 and 5.5,
    the instruction is%L%
1.3 5.5 add%L%
The numbers are placed on a stack, and the
"add" operator acts on and removes the numbers
from the stack, returning the result there.
%P%
It is necessary sometimes to place items
on the stack well before an
operator or procedure acts on them.
It can then be most unclear to the reader
just what an operator is doing.
This causes many PostScript
programs to be hard to read, and makes
debugging the program difficult. %P%As a
general comment, 90%% of programmers have not
worked with a language like PostScript before.
```

Care is required if % is needed in the text, as it is interpreted by Quikscript as the start of a layout instruction. To include a % within text, it should be doubled: %% , as in the example above. In this way, Quikscript can see that the % was really meant for output. If the text following a single % is clearly not a layout instruction, it will be output as the character unchanged by Quikscript.

The text typed needs to be broken into lines; it cannot be a stream of unbroken text as most word processors expect. Quikscript reads text in chunks up to the Return keystroke, and does not have the capacity to read chunks beyond 1024 characters in length.

## Bold and Italic

PostScript provides a number of fonts, which mostly have a choice of styles. *Italic*, or slanted, is selected in Quikscript by %IT%, **bold** by %BD%, and ***bold italic*** by %BD%IT%. Care is needed to remember to turn off the style selected. The opposite of italic is Roman, %RO%, and the opposite of bold is light %LT%. A style remains selected until turned off with the correct instruction.

For example, this typed text:

```
%BD%HMAS %IT%Voyager%RO% Disaster%LT%
%P%
The naval exercise was proceeding well on the
day of the accident. The aircraft
```

```
carrier HMAS %IT%Sydney%RO% was travelling
at cruising speed.
```

will be output as

### **HMAS *Voyager* Disaster**

The naval exercise was proceeding well on the day of the accident. The aircraft carrier HMAS *Sydney* was travelling at cruising speed.

## **Tabulation, or horizontal text placement**

If text is to be placed at a particular point in the current line, then the tabulation facility assists. In the simplest form, `%T,number%` will move to a new position on the page, *number* of millimeters from the normal left-hand margin on the current line.

For example, this typed text:

```
PostScript is a programming
language in which numbers or
parameters precede
the operator that acts on them.
Thus to add two numbers, 1.3 and 5.5,
    the instruction is%L%
%T,10%1.3  5.5  add%L%
The numbers are placed on a stack, and the
"add" operator acts on and removes the numbers
on the stack, returning the result there.
```

will be output as

```
PostScript is a programming language in which numbers or parameters
the operator that acts on them. Thus to add two numbers, 1.3 and 5.5, the
instruction is
    1.3 5.5 add
The numbers are placed on a stack, and the "add" operator acts on and
removes the numbers on the stack, returning the result there.
```

In interpreting the `%T,10%`, Quikscript will move to the position requested, regardless of what text has been placed on the line already. This instruction is useful for setting up simple tables; care is needed that the positions chosen allow enough width for the text content.

Leading blanks on a line are usually ignored, so in this example, the fifth input line has its leading blanks removed before the text on the line is added to the output. In all other places, blanks in your typed text are taken seriously.

If several lines of text are to share the same left alignment, a different instruction `%TB,number%` is used. Again *number* is the number of millimeters from the normal left-hand margin. In effect, this command sets a new left-hand margin for the following input text.

For example, this typed text:

```
The following instructions should be followed
in order to run Quikscript: %TB,5%L%
1. Create your document as a simple text
file.%L%
2. "New-line"/"Return" keystrokes will be
ignored.%L%
3. Add layout instructions%L%
4. Send Quikscript and your document to a
PostScript printer.%L%
%TB,0%Quikscript will process your document
within the printer,
interpreting any layout instructions.
```

will be output as

```
The following instructions should be followed in order to run Quikscript:
1. Create your document as a simple text file.
2. "New-line"/"Return" keystrokes will be ignored.
3. Add layout instructions
4. Send Quikscript and your document to a PostScript printer.
Quikscript will process your document within the printer, interpreting any
layout instructions.
```

It is important to remember to reset the left margin to its original position at the end of the indented text, the %TB,0% in this example. Another way of resetting the left margin is with the instruction %TB%, which means revert to the previous left margin setting. It is possible with this instruction to toggle between two states.

%TB sets the current left margin, and moves to that point on the current line. It is normally necessary to move to a new line if previous text is not to be overwritten, so a new line or new paragraph instruction will frequently accompany a %TB.

**Numbered points****Fractional line spacings**

The above example shows how short points can be presented effectively. However, if the points are longer, a combination of %TB for setting the new left margin for line continuations, together with %T for positioning particular details is useful. The gap between points can be made larger with an instruction %L,*number*%; *number* is the number of line heights to be moved down in addition to the normal movement. This can be fractional if necessary.

For example, to achieve this text layout:

The layout instructions are implemented using a table of three-element arrays.

1. The first element contains the layout command code, with multi-character codes preceding single character codes. This order prevents confusion between single character codes and multi-character codes commencing with the same letter.
2. The second element is a true/false value, indicating whether the layout code is "word-terminating" or not. This function will be explained below.
3. The third element is a PostScript procedure to be executed for that layout code.

we could type the text as:

```
The layout instructions are implemented using
a table of three-element arrays.%TB,10%
%L,.5%T,0%1. %T,10%
The first element contains the layout command
code, with multi-character codes
preceding single character codes.
This order prevents confusion between
single character codes and multi-character codes
commencing with the same letter.
%L,.5%T,0%2. %T,10%
The second element is a true/false
value, indicating whether
the layout code is "word-terminating"
or not. This function will be explained below.
%L,.5%T,0%3. %T,10%
The third element is a PostScript procedure to
be executed for that layout code.
%TB,0%%P%
```

The %TB,10% sets the left-hand margin, which will be obeyed when words must be placed on the next line. The %T,0% forces the point numbers to be placed at the original left margin, and the %T,10% causes text to begin at the same position as any line continuations. The %L,.5% causes a larger space between points, an extra half line height.

A similar layout can be achieved a little more easily with a special form of the new-paragraph instruction, as discussed in the "Paragraph layout" section below.

## Parameterized positioning

In the above example, the position "10" MM from the left margin recurs in many lines. It is possible to record this position once and reuse it many times without referring to the measurement again. The advantage of this is, if we later decide to alter the layout and change the indentation, only one number will need to be changed, rather than several. There is a command

```
%W%
```

which will record the current position ("Where"). That position can be referred to in %T or %TB instructions by

```
%T,?% or %TB,?%
```

If several positions are to be recorded, they are referred to by index: %W,1% stores the current position with index "1", and is referred to as %T,?1%. Up to fifty positions can be stored this way.

The above example can be expressed in this manner, giving identical output:

```
The layout instructions are implemented using
a table of three-element arrays.%TB,10%%W%
%L,.5%%T,0%1. %T,?%
The first element contains the layout command
code, with multi-character codes
preceding single character codes.
This order prevents confusion between
single character codes and multi-character codes
commencing with the same letter.
%L,.5%%T,0%2. %T,?%
The second element is a true/false
value, indicating whether
the layout code is "word-terminating"
or not. This function will be explained below.
%L,.5%%T,0%3. %T,?%
The third element is a PostScript procedure to
be executed for that layout code.
%TB,0%%P%
```

Then, if we decide to change the body indentation of the text to 15 MM, only the one number will need to be changed.

## New-line/Return taken seriously (eg Poetry)

There are many situations where we do not want the input text to be split up into words for packing into the output lines. Poetry, songs, program examples, and addresses are the most common. For this type of text, there is an instruction

```
%NF%
```

("No Fill") to take each line of input text and put it out on a separate line. It will still check to see whether the whole line will fit, and go to a new line if there is too much text. At the end of such text, we usually want to revert to the normal way of filling output lines with words. The instruction

```
%FI%
```

is used for this.

For example, this typed text:

```
%NF%
Joy to the world!  The Lord is come;
let earth receive her king;
let every heart prepare him room,
and heaven and nature sing,
    and heaven and nature sing,
        and heaven, and heaven and nature sing.
%FI%
```

will be output as

```
Joy to the world!  The Lord is come;
let earth receive her king;
let every heart prepare him room,
and heaven and nature sing,
    and heaven and nature sing,
        and heaven, and heaven and nature sing.
```

The last two lines had leading spaces in the original, which were retained in the output. In normal text processing mode, leading spaces on every input line are discarded, in contrast to the behaviour here in %NF% mode. The width of the space in the output was narrower than the original. This is because a space in most printing fonts is quite narrow, while most text preparation systems show a space the same width as any other character.

If we were including computer instructions or program listings in our document, we would probably choose to use a Courier (fixed-width) font, which shows better how the text characters line up. Font selection is explained below. Tab characters are generally ignored, so programs containing Tabs should be suitably filtered first.

If any of the input lines were too long to fit on a single output line, they would have been split at an inter-word gap, and continued from the beginning of the next line. It is frequently more desirable to have such continuations indented from the normal margin. This can be requested with an instruction

`%NF,number%`

where *number* is the number of millimeters of indentation desired on continuation lines.

A converse problem arises if we need to split the input line to prevent it being too long, possibly because of a large number of layout instructions. In other words, we want two input lines to be set on the one output line, within the context of all other input lines being placed on separate output lines. There is an instruction

`%\%`

that means: ignore anything that follows on this input line, but continue the current output line with the next input line.

## Headings Font Size Centring

When headings are needed, and in other situations, it is desirable to alter the size of the text. Size is expressed in "Printer's Points", of which there are 72 per inch. The default size is 12 pp, but any size, including fractional can be set. The instruction is

`%SZ,number%`

Headings are frequently wanted centred upon the page. An instruction

`%CL%`

takes the text following on the line and centres it between the current left and right text margins. Such a line would normally be followed by a `%L%` or similar instruction. Otherwise, following text would be set on the same line, following the centred text!

For example:

```
%SZ,15%%BD%%CL%Centred large heading%SZ,12%
%LT%
```

will be output as

### Centred large heading

It is possible to omit the number from a `%SZ%` instruction.

```
%SZ,15%%BD%%CL%Centred large heading%SZ%
%LT%
```

This is taken to mean, use the previous size. A short history of font-size changes is maintained, allowing several size-undo operations to be performed in this way. If only one change is recorded, then this becomes a toggle between the two states in the history.

The number given as font size is different from line height, the distance moved from one line to the next. This height is 1.17 times the font size by default (12 pp size gives 14 pp per line).

There is another instruction (%SN) for altering font size.

`%SN,number%`

differs from `%SZ,number%` in that it alters the height of text, but not the line spacing.

This is useful if text of a different height is to be mixed with other text *within a paragraph*. This happens fairly frequently in computing documentation, when including examples that are set in a Courier font within a paragraph of text in a more usual font, such as Times. The Courier text would appear very large and ungainly compared with the Times text, unless it was reduced in size.

However, we would not want to alter the inter-line spacing just for a small piece of embedded text in another font. If such text occurs at the end of a line, it would give uneven line spacing if `%SZ,number%` were used.

## Font Selection

PostScript printers provide a good range of fonts. In Quikscript, font is selected by name of a font family, with an instruction `%FN,name%`. *name* can be one of the following:

Tim (Times)	Hel (Helvetica)
Cou (Courier)	Sym (Symbol)
Pal (Palatino)	NewC (New Century Schoolbook)
Book (Bookman)	Ava (Avant Garde)
Cha (Zapf Chancery)	HelN (Helvetica Narrow)
Din (Zapf Dingbats)	Uft User-defined

The first four are guaranteed to be present in any PostScript machine. The user-defined font can be set up for a special purpose font, as is illustrated in some of the examples of embedded PostScript below.

Example of usage:

```
%FN,Hel%The quick brown fox died. Long live the
lazy dog.%FN%
```

will be output using Helvetica font as

```
The quick brown fox died. Long live the lazy dog.
```

A point to note in this example is the use of `%FN%` without a font family name. If the font family name is omitted, the font that was in use prior to the most recent change will be reverted to. This is useful if fonts need to be switched frequently, as when a fixed-width font such as Courier is wanted for computer examples. A short history of font changes is maintained, allowing more than one font-undo operation to be performed. If only one change is recorded, then this becomes a toggle between the two states in the history.

The default font is Times-Roman, %FN,Tim%, in Quikscript.

## Paragraph styles

There are more paragraph forms than those shown in the simple example above. Sometimes, we want to indent the first line. To achieve this paragraph layout:

This is an example of a common paragraph style. The first line of each paragraph is indented to make the paragraph beginning stand out. The paragraph body is flush with the normal left margin.

we precede the text with

%P,,5%

If we want to include a quotation, the paragraph can have its whole body indented:

Lewis Carroll has been quoted many times by mathematicians for comments such as:

*"Mine is a long and sad tale" said the Mouse, turning to Alice and sighing. "It is a long tail, certainly" said Alice, looking down with wonder at the Mouse's tail; "but why do you call it sad?"*

It is unusual for a mathematics professor to be able to bring pleasure to children, to think at so simple...

The quotation is preceded by a new-paragraph instruction

%P,10% %IT%

and followed by the instruction

%P,0% %RO%

Thus, the %P% instruction can include some numbers to control its layout. The general form is

%P,*body,firstline,ipgap,botgap*%

where

- |                  |                                                                                                             |
|------------------|-------------------------------------------------------------------------------------------------------------|
| <i>body</i>      | is the body alignment, in millimeters, relative to the current left margin as set by the last %TB in force; |
| <i>firstline</i> | is the indentation of the first line, relative to body alignment;                                           |
| <i>ipgap</i>     | is the change from the normal inter-paragraph gap;                                                          |
| <i>botgap</i>    | is the minimum distance that must remain between the start of the paragraph and the bottom of the page.     |

The indentation and alignment are measured in millimeters, and the change in the inter-paragraph gap and the distance from the page bottom are measured in line height units. Normally,

- paragraph alignment is at the current left margin,
- first-line indentation is zero,
- inter-paragraph gap is one blank line, and
- minimum page bottom distance is 30 MM.

To reduce inter-paragraph gap, the third number should be negative.

Any following %P% instructions will use the same numbers as were most recently specified on a %P instruction. If we give a paragraph specification

```
%P,,10,-.5%
```

then any following %P% instruction without numbers will adopt this same format.

Hanging first line style, as is useful for references or first word emphasis, could be set with

```
%P,10,-10%
```

For example:

```
%P,10,-10%
Adobe (1985a) %IT%PostScript Language
Reference Manual%RO%, Adobe Systems Inc.,
Addison-Wesley.
%P%Adobe (1985b) %IT%PostScript Language
Tutorial and Cookbook%RO%,
Adobe Systems Inc., Addison-Wesley.
```

will be output as

```
Adobe (1985a) PostScript Language Reference Manual, Adobe Systems
Inc., Addison-Wesley.
```

```
Adobe (1985b) PostScript Language Tutorial and Cookbook, Adobe
Systems Inc., Addison-Wesley.
```

In the above example the inter-paragraph gap is the normal single blank line. It is common for a smaller gap than this to be used in references or tabular point layouts. The instruction would become

```
%P,10,-10,-.5%
```

that is, the inter-paragraph gap is wanted to be half a line-height less than the default.

As we have seen earlier, the same sort of hanging first line could be achieved with use of a combination of %L%, %TB and %T instructions. The use of options on a %P% instruction is sometimes simpler.

We saw earlier how numbered points can be set; they can also be set using %P instructions:

```
The layout instructions are implemented using
a table of three-element arrays.%P,10,-10,-.5%
1. %T%
The first element contains the layout command
code, with multi-character codes
preceding single character codes.
This order prevents confusion between
single character codes and multi-character codes
commencing with the same letter.
%P%2. %T%
The second element is a true/false
value, indicating whether
```

```

the layout code is "word-terminating"
or not. This function will be explained below.
%P%3. %T%
The third element is a PostScript procedure to
be executed for that layout code.
%P,0,0,0%

```

This will produce the same output as was shown in the "Numbered points" section earlier. The advantage of using %P and %T to give the desired layout is that the position on the page is checked for each new-paragraph, and a new paragraph is not commenced unless there is at least a minimum amount of space at the bottom of the page.

The use of %T% without any indication of position has not been referred to before. Quikscript interprets this to mean go on the current line to the left margin plus any paragraph body alignment.

It is clear that %TB and %P can both set a left alignment for text. If both instructions are used in combination, the effect is for the %TB to set a left margin relative to the normal left margin, and the %P to set the paragraph alignment relative to the current left margin as set by %TB instructions.

## Vertical Placement

Just as it is possible to request a movement to a particular place across the line, it is also possible to move to a particular spot down the page.

```
%VT,number%
```

is the means of positioning at *number* millimeters from the normal first-line position at the top of the page. This can be useful for certificates or documents where precise positioning is wanted.

It is also possible to move vertically to a position remembered from before. If we have issued a %W% instruction in a document, then %VT,?% will return to the remembered vertical height. The horizontal position moved to is always the current left-hand margin. An example of its use will be shown in "Tabulation with Continuation" below. If there are several remembered positions, they can be distinguished, as before, by number, %W,1% to record position "1", and %VT,?1% to move there.

If space is to be set aside, perhaps for insertion of an illustration, whether by cut-and-paste or computer-generated, there is an instruction

```
%VM,number%
```

to move down the page by the *number* of millimeters. If there is not sufficient space at the bottom of the page, then Quikscript will go to a new page, and there move down the distance requested.

The distance to be moved can also be expressed as a certain number of line heights, by preceding the number by the letter "L". To move up the page by three lines, we would say

```
%VM,L-3%
```

In other words, if moving up the page (a negative number of lines), the sign must precede the number, not the "L".

### Headings at Page Bottom

This ability of `%VM,number%` to guarantee an uninterrupted block of space can be used to prevent section headings, major paragraphs, or tables from commencing too close to the bottom of a page. If we want a heading and part of the next paragraph, say 20 MM of text, to be unbroken, the input can be preceded by the instructions

```
%VM,20% %VM,-20%
```

The `%VM,20%` moves down by 20 MM, either on the current page if it will fit, or from the top of the next page. The `%VM,-20%` will move back to the start of the block, to the left margin.

An alternative method is to use a `%PH` instruction instead of a `%P` before the paragraph heading. `%PH` will ensure that a larger distance exists to the bottom of the page than a simple `%P`, sufficient for the header, another `%P`, and then a couple of lines of text. If insufficient distance exists, the text will be placed at the top of the next page. This feature of paragraphs is discussed in more detail above (Paragraph Style).

### Multi-columns

Most documents run simply down the page in a single column. Quikscript does allow multi-column output. There is an instruction

```
%NC,number%
```

which can be used to request the following text to be set into *number* columns.

It is sometimes desirable to commence a page with a single column, with heading and possibly with a summary across the page. Below this, the text is set in multi-columnar layout. The `%NC,number%` command enables this, in that the current point when the instruction is issued is remembered, and the second column is commenced at the same height on the page. Following pages will be fully multi-columnar. If you want to place text at the start of the next column, the `%NC%` instruction moves to this position.

**Page Margins**

The white space around the page of text can be set with a single command. By default, the left and top margins are 25.4 MM, and the right and bottom margins are 19 MM. These can be modified with

```
%PM,l,r,t,b%
```

where

*l* is the left margin,

*r* is the right margin,

*t* is the top margin, and

*b* is the bottom margin, all in millimeters.

Only those numbers to be changed from the existing setting need to be specified, eg.

```
%PM,,19,%
```

would reduce the top margin to 19 MM.

**Column Width**

The overall width of the set text can be defined with a command,

```
%CW,number%.
```

*number* is the width in millimeters. By default, the column width is determined from the page dimensions, the width of the page margins, and the number of columns of output per page. This can be locally adjusted with the %CW command. The former width can be reverted to with %CW%.

By default the page width of an A4 page is 165.6 MM. A width of 161 MM will give a right margin the same width as the left.

**Special Characters**

PostScript fonts contain many characters other than the normal keyboard characters. They can be output in Quikscript with a

```
%C,number%
```

instruction. The *number* can be chosen by looking in the table in Appendix C.

For example:

```
%C,183% %TB,10%The monitor cost %C,163%200
%C,208% $500 %C,208% but could have been
bought for %C,165%40000 in Japan.
```

will be output as

- The monitor cost £200 — \$500 — but could have been bought for ¥40000 in Japan.

It is possible by this means to distinguish between the three characters -, −, and —, respectively hyphen, minus, and dash. The first is obtained with the simple minus keystroke, the second with %C,177%, and the third with %C,208% in the normal PostScript character set. See Appendix C for the complete set.

It is also possible in this way to create European characters that are not in the usual character set. This will be illustrated below in an example ("European Characters") and in Appendix C.

### Gaps not to be broken

Quikscript takes the input text, and looks for spaces to break between words, each word being fitted separately into the current output line. If it does not fit, it is placed on the next line. Sometimes, we do not want a word break at particular gaps. Common examples are in telephone numbers, credit card numbers, or between initials and a person's surname. To prevent such breaks from occurring, they can be represented to Quikscript with the notation `%_%`, as for example in

`258%_7734`

Longer spaces can be set aside with more underline characters between the `%` characters.

### Page Numbering

Pages can be numbered, if desired, with the instruction `%PN%`. The page number will be placed at the upper right-hand corner of the page, preceded by the text, "Page". Page numbering will commence on the next page, number "2". If page numbering is to commence at a different number, the number can be given in `%PN,number%`, in which case the *number* will appear on the current page.

More sophisticated page numbering or annotations are described below in "Special page annotation and numbering".

### Superscripts, Subscripts, Mathematics

Technical documents often need superscripts or subscripts, to represent a number raised to a power or an instance of a set, for example. The instruction for subscript is `%V%`, for superscript `%^%`, and to revert to normal size `%=%`.

For example:

`A%V%i%^2%=%`

will be output as

$A_i^2$

Many mathematical symbols are available in the Symbol font (Sym), as shown in Appendix C. To represent a summation

$\sum_{i=1}^8 ( \text{Max}(0, \text{Slope}_i^p) )$

we would give the instructions

`%IT%FN,Sym%S%FN%W,1%^8  
%T,?1%V%i=1%=( Max(0, Slope%V%i^p% ) )%RO%`

Although this may be somewhat daunting, piece by piece it is very simple. The whole line is set as italic (`%IT%`, with `%RO%` at the end), because mathematics often looks best this way. We change font to Symbol (`%FN,Sym%`) and output the character "S", which in that font is the Greek  $\Sigma$ . We must revert to the normal font for the remaining output (`%FN%`). After the summation sign, we want a subscript (`i=1`) and superscript (8), and we want them at the same horizontal position. So we record the position (`%W,1%`) and return to it (`%T,?1%`) after putting out the superscript (`%^%8`). At the end of the subscript (`%V%i=1`) we need to revert to normal size and placement (`%=%`).

The only tricky part of this example is the need in this case to represent the superscript before the subscript. There is a general rule that if we want to have both superscript and subscript with the same horizontal alignment, then we must first request the *shorter* of the strings. Otherwise, following text could overwrite earlier output.

### Full Justification, Centred, Right Justification

Quikscript allows you to output text fully justified (ie. right and left ends of lines aligned), centred, or right-justified, as well as the default behaviour of left-justification. This default was chosen because of the firm belief that ragged-right text is easier to read, although less tidy, than fully justified text. Furthermore, if text is output in a narrow column, the allocation of white space between words to pad out the right margin can be quite ugly. This can be alleviated to a limited extent by allocating some of the spare space between characters of words, or by hyphenating long words. Neither of these is currently done in Quikscript.

The layout instructions to select the method of text justification are: `%FJ%` for full justification, `%LJ%` for left-justification, `%CT%` for centred, and `%RJ%` for right-justification.

For example:

Quikscript allows you to output text fully justified (ie. right and left ends of lines aligned), centred, or right-justified, as well as the default behaviour of left-justification. This default was chosen because of the firm belief that ragged-right text is easier to read, although less tidy, than fully justified text. If text is output in a narrow column, the allocation of white space between words to pad out the right margin can be quite ugly. This can be alleviated to a limited extent by allocating some of the spare space between characters of words, or by hyphenating long words. There is no hyphenation facility currently in Quikscript, but space is sometimes apportioned between letters of words to space them out.

is set with `%FJ%` preceding the text. With `%CT%` instead of `%FJ%`, it would look like:

Quikscript allows you to output text fully justified (ie. right and left ends of lines aligned), centred, or right-justified, as well as the default behaviour of left-justification. This default was chosen because of the firm belief that ragged-right text is easier to read, although less tidy, than fully justified text. If text is output in a narrow column, the allocation of white space between words to pad out the right margin can be quite ugly. This can be alleviated to a limited extent by allocating some of the spare space between characters of words, or by hyphenating long words. There is no hyphenation facility currently in Quikscript, but space is sometimes apportioned between letters of words to space them out.

If full-justification is being used and a block of text is to be set using %NF% (do not fill lines, see "Not filling lines" above), this will override the full-justification until %FI% is encountered.

There is a common need to centre or right-justify a single line or piece of text, and separate instructions are provided for this. To centre a line, the %CL% instruction was described above. To right-justify a line of text, the %RL% instruction should be used preceding the text.

Text can be right-justified to a certain position within a line, with %RL,*pos*% where *pos* is the distance from the normal left margin. Remembered positions can also be used with the "?" notation.

This feature can be useful in simple tables of numbers:

20	400	2
5000	7	7932

would be set as

```
%T,20%%W,1% %T,40%%W,2% %T,60%%W,3%
%RL,?1% 20 %RL,?2% 400 %RL,?3% 2
%RL,?1% 5000 %RL,?2% 7 %RL,?3% 7932
```

**Dot Fill**

There are times such as in a table of contents when the space between two pieces of text is wanted to be filled with a repeating character, such as dots. This is provided with

```
%DF,pos,ch%
```

where *pos* is the horizontal position on the current line from the left margin that is to be filled to, and *ch* is the character to be used for filling. If *pos* is omitted, fill occurs to the end of the line. If *ch* is omitted, the normal full-stop (period) character is used.

For example, the output

```
1. Introduction ..... 3
2. Simple Quikscript examples ..... 15
```

is achieved with the instructions

```
%NF%
1. Introduction %DF,150%%RL%3
2. Simple Quikscript examples %DF,150%%RL%15
%FI%
```

Dots can be filled to a position remembered from earlier in the document using the notation `%DF,?2%`, for example, if the position was saved earlier with a `%W,2%`.

The character can be the actual keystroke, or a decimal numeric code as used as used for special characters.

**Tables**

The simplest way a table can be built in Quikscript is to turn off the normal line-filling behaviour, by setting `%NF%` as described in the earlier section on "not filling lines". Columns of text can be aligned by use of `%T,position%` before each piece of text. The position can be a distance in millimeters from the normal left margin, or can be a remembered position set up at the start of the table as described in the section "parameterized positioning" above. An example of this is shown in a earlier section entitled "right-justified text".

If the text of a cell could span more than one line, or if we want boxes to be drawn around the cells, this simple approach is cumbersome and inadequate. A more powerful scheme is available using the instructions `%ST,...` to start a table, `%VL...` to indicate where vertical lines are to be drawn, `%HL...` for horizontal lines, `%NR%` to commence a new row of the table, `%CO%` to commence a new column in this row, and `%ET%` to end a table.

Fruit	Amount	Timing	Directions
Apples	2 spoonfuls per litre	Early June	Do not spray after bud burst.
Peaches	4 spoonfuls per litre	May or June	May be mixed with winter oil in the early part of the spraying season.
Nectarines	2 spoonfuls per litre	First spray in early June. Second spray must not follow bud-burst, and may be combined with an aphid treatment.	

This table can be created with the instructions:

```
%ST,45,65,90,115,160%%VL,1,1,2,3,4,5,5%%HL2%
Fruit %CO%Amount %CO%Timing %CO%Directions
%HL%%NR%
Apples %CO%2 spoonfuls per litre %CO%Early June
%CO%Do not spray after bud burst.
%HL%%NR%
Peaches %CO%4 spoonfuls per litre %CO%May or June
%CO%May be mixed with winter oil in the early
part of the
spraying season.
%HL%%NR%
%CO,1%Nectarines %CO,2%2 spoonfuls per litre
%CO,3,4%First spray in early June. Second spray
must not
follow bud-burst, and may be combined with an
aphid treatment.
%HL2%%ET%
```

The %ST,45,65,90,115,160% instruction (start-table) establishes where the column margins are. The numbers indicate distance in millimeters from the normal left margin. Remembered positions can be referred to using the "*number*" notation described in "parameterized positioning" above. The number indicating the right-hand margin of the table can be omitted, provided that the trailing comma is present, and the page margin will be assumed implicitly. Eg. %ST,45,65,90,115,%

The table is ended with the instruction %ET%.

%VL,1,1,2,3,4,5,5% determines which of the column margins will have vertical lines drawn, and how thick they will be. Duplicating a margin number doubles the line thickness.

%HL2% will cause a horizontal line of double thickness to be drawn across the table. Normal horizontal lines are produced by %HL%.

Text is placed in different columns of the current row in sequence, with %CO% causing a move to the next column. A column can be chosen explicitly by referring to the column number, as is done in the "Nectarines" row with %CO,2%. The last cell in this line is forced to span two columns width by giving a column range %CO,3,4%.

Note that row depth is determined automatically by the content of the deepest of the cells in that row.

Product	Price	
gnats	gram	\$13.65
	each	0.01
gnu	stuffed	92.50
emur		33.33
armadillo	frozen	8.90

This more artificial example illustrates control over where horizontal lines are placed, and control over justification of the columns. It is produced by:

```
%ST,55,80,100,120%HL2%VL,1,1,2,4,4%TJ,lcr%
Product %CO,2,3%CL%Price
%HL%NR%
gnats %CO%gram %CO%$13.65
%HL,2,3%NR,2%
each %CO%0.01
%HL%NR%
gnu %CO%stuffed %CO%92.50
%HL,1%HL,3%
%NR%
emur %CO,3%33.33
%HL%NR%
armadillo %CO%frozen %CO%8.90
%HL2%ET%
```

The text in the second column is centred while the text in the third is right-justified. The instruction %TJ,lcr% establishes this table justification. The lower case characters "l", "c" and "r" mean respectively left-justified, centred, and right-justified. "f" can be used to request full justification, though this is likely to produce ugly results unless a column is fairly wide. As we saw in the first example, left justification will be used if no specific rule is requested in a %TJ instruction.

The column justification can be overridden for a particular cell, as was achieved with the "Price" heading, which would have been right-justified if the %CL% had been omitted before the word.

The "gnats" row has a horizontal line beneath that spans the second and third columns only. This was produced by %HL,2,3%. Similarly, where we did not want a line beneath "stuffed", two line segments were requested with %HL,1%%HL,3%. A single number following "%HL," indicates the column that is to have a horizontal line, whereas two numbers indicate a column range.

Vertical lines do not need to be drawn at all column margins; the third margin is omitted from the %VL list by omitting the number "3", eg %VL,1,1,2,4,4%.

Professional typesetters use vertical and horizontal lines sparingly in setting tables, in contrast to computer users more experienced with spread-sheets than in visual design. Many drawn lines tend to have a heavy and formal appearance. Compare the table below with the earlier example.

	Amount	Timing	Directions
Apples	2 spoonfuls per litre	Early June	Do not spray after bud burst.
Peaches	4 spoonfuls per litre	May or June	May be mixed with winter oil in the early part of the spraying season.
Nectarines	2 spoonfuls per litre	First spray in early June. Second spray must not follow bud-burst, and may be combined with an aphid treatment.	

These table features will not handle very well large tables that span more than a page. If text within a cell must spill onto a new page, then subsequent cells in the same row will commence on the new page, rather than being similarly split across the page boundary.

Most of the text of this document has been set as a table by a different method, using a combination of the %TB , %W , %VT and %CW instructions. Each section contains a title in the narrow left column, and the descriptive text on the right. At the start of each new section, the position is remembered (%W), the left margin is fixed with %TB, the right margin with %CW, and the heading is then placed. To place the right-hand column, we change the left and right margins again using %TB and %CW, and set the vertical placement of the first line using %VT to return to the remembered position. This is a workable scheme, though is more cumbersome than the simpler scheme described above, which automates the margin setting. This approach has been used because there is no way in Quikscript to place a table within a table.

**Page Header**

If pages are to be output with page numbers (see %PN%), there will be a short piece of text printed beside the number at the top of the page. By default, this will be "Page ".

The text beside the page number can be changed using two layout markers, %SH% and %EH%. %SH% indicates the start of header text, and %EH% marks the end of header text. Multiple words can be included between these markers, and it is possible to include Quikscript layout marks such as %C for special characters. It is not possible to include PostScript or line-breaking layout marks within the text.

For example,

```
%SH%Special Features %EH%
```

would cause this and following pages to have page numbers printed as:

```
Special Features 2
```

**Text Colour**

The colour used for text output can be set with the layout marker  
%TC,*red,green,blue*%

The numbers for *red, green, blue* should be in the range from zero to one. To revert to normal black text, %TC% should be used.

**Background Colour**

There is restricted support for a background colour for text in a document. This is intended for use in highlighting a paragraph or block of text. The layout marker takes the form:

```
%BG,red,green,blue,left,right%
```

The numbers for *red, green, blue* should be in the range from zero to one.

The final two numbers indicate the left and right boundaries of the background. The *left* will be subtracted from the normal left margin of the text, and *right* will be added to the normal right margin. Both are measured in millimetres. If these numbers are omitted, the default values for them are 14 and zero printers points respectively.

To restore the normal background, use

```
%BG%
```

It is not suitable for providing a background to a piece of text within a paragraph; it fills an area from the left to the right margin of the column or page. It will not work very well if there are large

changes in font size within the block of text. It will not provide a background for illustrations that might be included. If a table is part of the text requiring a background, any horizontal cell borders will probably be lost by the filling operation. There is a separate facility to fill the background of a table (%BC below).

If a whole document is to be prepared with a background colour, it would be simpler to modify the new-page PostScript procedure /Eject as described below, filling a rectangular area of each new page before text is placed.

**Table background colour**

It is possible to specify a background colour for selected cells or columns of a table. The marker is

```
%BC,col1,col2,red,green,blue%
```

The first two numbers indicate the range of columns that are to have their background filled. If only a single column is to have a background, then the same number must be repeated.

Colour is given by three numbers in the range from zero to one.

To finish using the background colour for a particular column range, use

```
%BC,col1,col2%
```

To finish using background colour in the table, simply use

```
%BC%
```

The main caution with selectively using colour in a table is that colour should be turned on immediately after %NR or %CO, and turned off before %NR or %CO. Also, background colour in a table will not be handled correctly in some situations where there is a variety of font sizes in a cell of a table.

Example:

Fruit	Amount	Timing	Directions
Apples	2 spoonfuls per litre	Early June	Do not spray after bud burst.
Peaches	4 spoonfuls per litre	May or June	May be mixed with winter oil in the early part of the spraying season.
Nectarines	2 spoonfuls per litre	First spray in early June. Second spray must not follow bud-burst, and may be combined with an aphid treatment.	

is created with these instructions:

```

%ST,45,65,90,115,160%%VL,1,1,2,3,4,5,5%HL2%
Fruit %CO%Amount %CO%Timing %CO%Directions
%HL%%NR%%BC,1,1,1,.8,.9%BC,2,4,.8,1,1%
Apples %CO%2 spoonfuls per litre %CO%Early June
%CO%Do not spray after bud burst.
%BC,2,4%
%HL%%NR%
Peaches %CO%4 spoonfuls per litre %CO%May or June
%CO%May be mixed with winter oil in the early
part of the spraying season.
%HL%%NR%
%CO,1%Nectarines %CO,2%2 spoonfuls per litre
%CO,3,4%First spray in early June. Second spray
must not follow bud-burst, and may be combined
with an aphid treatment.
%BC%%HL2%
%ET%

```

If you need to change font size within a table where background colour is being used, it might help to understand how Qs works in order to find the set of instructions to do what you want.

When the %BC instruction is encountered, a shaded stripe is drawn at the top of the table in the selected columns. The height of this stripe is determined from the current font size. Each time a new line is needed in a cell, a coloured stripe is drawn before any text is output, and again, the position of the top and bottom of this stripe is based on the current text height.

If you are going to change font size, it may be safest to first go to the line position where the text will be output, change the size, and then output the text in that line, restoring the original size before going to the next line.

## Table of Contents

If a document is to be added to over time, it is convenient to have Quikscript generate the table of contents. It is necessary to place markers and text within the document body, indicating what is to appear in the contents. Quikscript will add the page number to the text, recording it for subsequent output with all the other items.

An item for the contents is indicated with the markers %SC% and %EC%, eg:

```
%SC%Chapter 4: Colour separation%EC%
```

This text will not appear in the document at this point, but will be held over for later output. The text may contain Quikscript layout marks. These will have no effect locally within the document, and are only acted on when the table of contents is printed.

Each contents line will have appended to it the text

```
%DF, ?%RL%
```

and the page number. The DF and RL layout marks are intended to improve layout of the table when it is printed. For these to work properly, it would be necessary to tab to a suitable position for the end of the dots, and record it with, for example,

```
%T,150%%W%
```

before printing the contents.

The page number recorded in the index may have a chapter abbreviation in front of it, if chapters or appendices are numbered separately from one. This chapter abbreviation can be set up by eg:

```
%CA,A-%
```

which will cause all page numbers following this point to commence with "A-" when they are recorded in the index or table of contents. No blanks are allowed in this abbreviation.

At the end of the document, the table of contents could be printed with the following:

```
%BD%%SZ,16%%CL%Contents
%SZ%%LT%%P%
%TB,20%%CW,140%%T,130%%W%%T%
%PC%
%TB%%CW%
```

The first two lines put out the heading in an appropriate style. The third line sets the left margin (%TB,20%) and the right margin (%CW,140%) so that the table is not too wide. Each line of the table will have a dotted line going to a point previously recorded using a %W%, so the remaining instructions on the third line record a position 130 MM from the normal left margin for this dotted line. The table of contents is printed with %PC%. Finally,

the left and right margins are restored to their previous values (%TB%%CW%).

The line above could be printed as:

Chapter 4: Colour separation ..... A-37

The printed contents page will need to be moved to the start of the printed document. Its page number will be wrong unless an extra instruction is given before the %PC% to set the page number correctly. If no page number is needed on the contents page, we would give:

```
%PN, 0%
```

If a large document is being written with chapters separately prepared, then it is possible to output the table of contents in parts to the screen or to "standard output" if a PostScript interpreter is used. These parts would be collected from the separate chapters or sections of the document, joined together into a single file, and printed separately. The command to output to the screen is:

```
%OC%
```

## Index

An index to a lengthy document can be built by including layout marks and appropriate text near the section to be indexed. This text will be recorded, and can be output at the end of the document along with page numbers, all neatly sorted alphabetically.

To specify an item for the index, the markers %SI% and %EI% are used around the text:

```
%SI%Succulent plants%EI%
```

This text will not be output at the current point. It will be held for later printing or output to the screen.

The page number may have a chapter abbreviation in front of it, if chapters or appendices are numbered separately from one. This chapter abbreviation can be set up by eg:

```
%CA, 3-%
```

which will cause all page numbers following this point to commence with "3-". No blanks are allowed in this abbreviation.

To print the index at the end of the document, the following set of instructions would typically be given:

```
%NP%%BD%%SZ, 16%%CL%Index
%SZ%%LT%%P%
%NC, 2%
%PI%
```

```
%NC,1%
```

The first two lines put out a suitable header. The third line makes the remainder of the page be printed using two columns. The fourth line

```
%PI%
```

causes the index to be printed.

Text can contain layout marks. These are not acted upon until the index is to be printed. Be aware that the sorting of the index is based on the raw text, and that layout marks are treated as text in the sorting process.

If the document was being built up with sections as separate documents, the index for each section could be output to the screen or to "standard output" for merging into a single index. This can be done with the instruction

```
%OI%
```

It is common for the same text to appear in the table of contents and in the index. Quikscript accepts the one text description embedded within both index item and contents item markers:

```
%SC%%SI%Egyptian hieroglyphs%EI%%EC%
```

A single buffer is used to hold the text being recorded for later output, so it is important that the index and contents text have a common starting point. Do not put any text between the %SC% and the %SI%.

The index is built in sorted order. Sorting is based on the ASCII representation of characters, which is not always the desirable sorting order. A major disadvantage is that all upper-case letters precede lower-case letters. This means that a word such as "HTML" will precede "applet" in the index. It is not easy to fix this in Quikscript, without greatly enlarging the program.

There are two approaches you can adopt to avoid this problem. In the first approach, when indicating the text that is to appear in the index, always start with an upper-case letter. This will not give ideal sorting of the remaining characters in the words, but will at least group items sensibly based on the first character.

In the second approach, the index would be output rather than printed, with the output collected into a file. That file could be sorted ignoring case of the text using some sort package on your computer, and output using Quikscript in a subsequent step. This approach may already be necessary if a document is split into separate files, with the index for each file merged into a single sorted file.

A program is provided to assist with index sorting. See the description of "sortIndex" in Appendix B.

## More Sophisticated Applications Embedding PostScript

Provision is made in Quikscript for PostScript code to be included in the text input itself. One application for this is to enable Quikscript to be modified from within a document being formatted by it. In essence, Quikscript can be self-modifying, because PostScript code is interpreted rather than compiled. Another reason for doing this is to gain access to special PostScript features, such as modified fonts or rotated page layouts, that will be illustrated in the examples that follow.

Two constructions can be used for including PostScript instructions in a document. The first is the simple form:

```
%PS embedded PostScript %
```

The whole instruction must fit on the one line. It is restricted further in that it cannot use string variables, that is, text within parentheses, with the simple *def* PostScript operator.

The alternative is:

```
%PS%  
embedded PostScript  
PE
```

This construction is unrestricted in what PostScript may be used, except that it must execute without error. It does not have automatic access to internal Quikscript functions or variables in the same way that the first construction does.

### Double Spacing

If a document were needed to be printed double spaced, perhaps to meet the requirements of a publisher, the command:

```
%PS /lIGap 2 store%%SZ,12%
```

would effect the change. Paragraph specifications might need to be altered to reduce the inter-paragraph gap, which will also be almost doubled by this instruction. The default value for parameter *lIGap* is 1.17; this establishes the distance between lines based on the font size. If the font size is 12 pt, an *lIGap* of 1.17 will give the distance between baselines of 14 pt (12 x 1.17). Any interline spacing can be set by changing this number.

A size specification must follow the PostScript for the change to take effect.

### Special page annotation and numbering

As was discussed earlier, the text that usually accompanies the page number, "Page", can be changed to something more specific to the document using %SH% and %EH% to bracket the text. This text can include a limited set of layout marks that affect the font. Before this text is displayed, a header prefix set of commands is executed, which by default is

```
(%FN,Tim%%SZ,12%)
```

The font or text size can be changed by modifying this string, eg.

```
%PS%
QSdict /HdrPre (%FN,Hel%%SZ,10%) put
PE
```

Page numbering is performed by two routines in Quikscript:

```
%PS%
QSdict begin
% - PrPagNum -
% Increment and print page number
/PrPagNum
{ IncPagNum PagCnt 1 gt NumberIt and
  { gsave WordWid SaveTempState 0 mark
    HdrPre UseKwds pop pop
    /PagNum PagCnt TmpS cvs store
    % convert page number to integer
    PagNumPos PagStr showStr PagNum show
    RestoreStates pop pop RestoreTempState
    /WordWid exch store grestore} if
  } def

% - PagNumPos -
% Move to the position for page number.
/PagNumPos
{ RgtBnd PagNum stringwidth pop sub PagStr
MeasureStr sub
  PagLen TopMrg sub HdrSpa add moveto } def
end
PE
```

The first routine establishes whether the page number is wanted, and outputs it on the page. The second routine determines the position for the start of the page number text. These routines can be changed if you want page numbers to be placed in different positions, or if you want several text items, either as running headers or footers to the page.

### Centred Page Numbers

If page numbers were wanted centred on the page, the routine for moving to the position for page numbering would need to be revised. The PostScript instructions in *cntpagnm.qs* should be included following Quikscript and before the document. This will put the page number centred at the bottom of the page, and the page heading text (PagStr) centred at the top of the page. (See %SH and %EH for setting the header text.)

Note that it is still necessary to use the %PN% directive to request page numbers to be included in the page.

**Left/Right Page  
Numbering**

Documents that are to be reproduced double-sided ideally need the page number to be on the outside of the page, ie. on the right-hand side of odd numbered pages and on the left for even-numbered pages. To achieve this in Quikscript, the two procedures would need to be replaced by including the PostScript instructions in file *altpagnum.qs* between Quikscript and the document.

Note that it is still necessary to use the `%PN%` directive to request page numbers to be included in the page.

**Landscape  
orientation**

If a document is to be printed sideways on a page, the PostScript instructions in *rotate.qs* should be included after Quikscript and before the document.

**Layout marker  
character**

If the character that delimits Quikscript commands, `%`, is inconvenient, another may be substituted. To change to using `#` instead of `%`, the command would be:

```
%PS%
  QSdict /Esc (#) put
PE
```

A one-line `%PS` instruction cannot be used because a text string is involved. This constraint arises from the way composite objects are handled on the stack.

**Graphic Page Border**

To place a border around every page, an internal routine in Quikscript, *PrPagNum*, can be redefined. The normal version of this routine is given above in "Special page annotation". In this example, the organization name is to be placed at the bottom below the border, and the section within the organization will be placed at the top of the page, also outside the border:

```
%PS%
/Tit1 (University of Woolloomooloo) def
/Tit2 (Department of Leisure) def
/PagNumPos{295 780 moveto} def
QSdict begin
/PagStr() def
/PrPagNum
  { IncPagNum gsave WordWid SaveTempState 0 mark
    % put a nice border around the page
    newpath 60 36 moveto
    535 60 24 -90 0 arc 535 780 24 0 90 arc
    60 780 24 90 180 arc 60 60 24 180 270 arc
    stroke
    HdrPre UseKwds pop pop
    % convert page number to integer
    /PagNum PagCnt TmpS cvs def
    PagNumPos PagStr showStr PagNum show
    65 26 moveto Tit1 show
```

```

        65 808 moveto Tit2 show
      RestoreStates pop pop RestoreTempState
      /WordWid exch store grestore
    } def
  end
PE
%CW,161%

```

Clearly, any type of graphics can be placed on every page in this way. The graphics are placed just before the page is output. In those rare cases where the graphics must be placed before any text on the page (such as for the word DRAFT in large grey letters across the page), then the "Eject" routine would need to be altered instead, or the "showpage" operator redefined (see the file "draft.qs").

## European characters

PostScript fonts include drawing instructions for most European characters, but these are initially inaccessible. If some of these are to be used, the Encoding Vector of the current font must be altered to add reference to them. There are several ways this could be done.

The approach we adopt in *pdffnt.qs* or *isolatin2.qs* is to redefine some of the characters given in Appendix C so that the number corresponding to the European characters matches the ISO Latin-1 or ISO Latin-2 encoding. This is consistent with the ISO standard, except that the Latin-1 set also contains all of the common typesetting characters. The special characters are shown in Appendix C, along with their code numbers. This example is based on Program 18 of the original PostScript blue book.

The file *pdffnt.qs* or *isolatin2.qs* must be sent to the printer following *Qs* and preceding the text to be output. The European characters would be selected using *%C,code%* within the input text, or by the document containing the 8-bit code for the Latin-1 or Latin-2 character. Eg.

Brückner

would be requested using *Br%C,252%ckner*. The number to use can be determined by inspecting the numbers and character names in the PostScript text in *pdffnt.qs*. The appearance of the characters is shown in Appendix C.

The PostScript program *pdffnt.qs* or *isolatin2.qs* should typically be included at the top of a document, following the Quikscript header. It is necessary to declare which font is to be used in your document *after* this program. All fonts (except for *Sym* and *Din*) will be affected by this program, so that switching between fonts will retain a consistent character representation.

### Adding a new font (Outline)

It is possible in PostScript to add a new font, or define a new font based on an existing font. An example is given in the PostScript blue book of an outline font, with the centre of the characters white rather than filled. Quikscript has been set up with an extra, undefined font family for you to add an extra one of your own. It is referred to in a document with the instruction `%FN,Uft%` (for user-font). If we want to set up this font to be the outline font given in the blue book, we would include the PostScript from *outlnft.qs* in our document.

This example defines bold, italic, and italic bold versions of the font, so that all Quikscript instructions may be used with the font. The Helvetica font was used here, but other fonts could have been used if the explicit font name references in the last seventeen lines of that file are replaced with the corresponding names for the desired font.

Text can be output with the font in this way:

```
%SZ,14%%FN,Uft% Outline font text%FN%%SZ%
```

which will print as:

Outline font text

### Adding a new font (Small Capitals)

Another example of adding a new font arises from the need to represent some words in capitals. If normal capitals are used, the words stand out in the surrounding text because capitals are generally wider than lower case letters. In typesetting, this is often overcome by using a smaller font for such letters. This is possible in Quikscript, but if several words are to be output, there is the risk that a new line break will come in their midst, and the line gap will be inappropriate for the surrounding text. The `%SN` instruction can fix the line gap problem, but this can still cause problems in that such an instruction within a word will act as word end as far as Quikscript is concerned.

A better way is to define a new font based on the current font, in which lower case letters are represented with smaller capitals than the normal upper case letters. This will allow large capitals to be intermixed with the smaller capitals, as well as overcoming the word-break problem.

The instructions to set up the "Uft" font for small capitals is given in file *smallcap.qs*. This acts on the font that is current when it is executed, so if a font other than Times is to be set in small capitals, that font should be selected first before the *smallcap.qs* instructions are included. The small capital font can be selected at any place in the following document using the instruction `%FN,Uft%`.

Using this example, the text

## POSTSCRIPT

would be achieved by embedding the program segment in *smallcap.qs* in the document *after* the initial font selection, and representing the text with `%FN,Uft%PostScript%FN%`.

### Adding a new font family

The names of fonts in Quikscript are set in two tables: "Font" contains the PostScript names of the fonts that are available, and "FontNam" contains the abbreviations by which they are known in a Quikscript document. The names in Font come in groups of four for the different styles, and there must always be four font names for each family, whether there are actually four fonts or not. At the end of these tables there are three spare slots for abbreviations, named Uft, U2 and U3, which have corresponding dummy names in the Font table. To install one of the fonts described above in Quikscript, the names in the table are replaced with the newly created font names.

Quikscript contains a procedure, "AddFont", which simplifies registering a Quikscript abbreviated name and the associated internal font names. This procedure adds the names to the "Font" and "FontNam" tables.

For example, a family that is widely used is called CharterBT. If this family was to be given the abbreviation "Cht", the following PostScript might be used at the head of a document:

```
%PS%
QSdict begin

    (Cht) /CharterBT-Roman /CharterBT-Italic
          /CharterBT-Bold /CharterBT-BoldItalic
    AddFont
end
The actual font definitions are put here
%include fonts/Charter-Bold%
%include fonts/Charter-Roman%
%include fonts/Charter-Italic%
%include fonts/Charter-BoldItalic%
PE
```

The internal names of the fonts must be found from within the font files, which must be included in the document to be printed. The program *catex.c* (see below) is a useful tool to pull the font files into the document only when they are required.

**Embedded graphics**

Graphics can be output. To place a solid horizontal line from the current position:

```
%PS%
gsave 80 MM 0 rlineto
.5 MM setlinewidth stroke
grestore
PE
%L%
```

is a simple example. After a %L%, it will produce:



Any PostScript can be included in this way. If the embedded PostScript leaves a current point in the graphics state (such as by gsave/grestore as in the example above or by an explicit moveto at the end), that point will be used for placing following text on the page. If there is no current point, the last current point known to Quikscript will be used for following text.

It is important that embedded PostScript should not remove any items from the stack. Some state information is held on the stack for when Quikscript resumes. New items may be left on the stack by the embedded Postscript, and Quikscript will simply discard them.

**External Illustration**

In general it is possible to include any PostScript drawing instruction anywhere in a Quikscript document. The most common need is to insert an illustration that has been created with some external drawing package or image scanner.

The general procedure is to move to a suitable place on the page, move the origin so that the illustration does not interfere with text on the page, output the illustration, and then restore normal Quikscript text processing.

The general form is:

```
%VM,100%%PS%
gsave currentpoint translate
    place the PostScript illustration here
grestore
PE
```

The distance moved with the %VM would have to be sufficient for the size of the illustration. Also, the "currentpoint translate" assumes that the illustration is set up to appear at the lower-left corner of a page. Any other placement would require another "translate", which may have to be established by trial and error.

It is important that PostScript instructions included in this way be "encapsulated PostScript", that is, instructions that are capable of being embedded within another document. This PostScript is not allowed to include instructions that have irreversible effects on the state of the current page being output, such as "initgraphics" or "erasepage". It is common for packages generating illustrations to include a "showpage" at the end. This is not in breach of encapsulated PostScript rules, for this makes it possible to print an EPS file by itself. However, it assumes that either the call to "showpage" at the end of the illustration must be removed, or "showpage" must be disabled before including the illustration in a document. In the latter case, "showpage" would need to be re-enabled after the illustration for Quikscript to continue to work properly.

```
%VM,100%%PS%
/QsEPSSave save def  currentpoint translate
/showpage {} def
    place the unmodified EPS illustration here
QsEPSSave restore
PE
```

Because this arises so often in document preparation, a couple of procedures have been added to Quikscript to reduce typing in a document:

```
%VM,100%%PS%
EPSstart
    place the unmodified EPS illustration here
EPSend
PE
```

#### Guidelines for Embedding PostScript

With the construction  
`%PS%`  
*embedded PostScript*  
`PE`

the PostScript interpreter itself will commence reading input from the start of the line following the `%PS%`. Any text following `%PS%` on the same line will be ignored. `PE` is defined as a PostScript procedure to reinvoke Quikscript. It must not be followed by any other text on the same line.

It is possible to redefine any of the Quikscript procedures or values, intentionally or accidentally. If only graphic output is intended without any side-effects, it is safest to bracket your PostScript with "gsave" and "grestore" commands. This will ensure that the current position is not lost in the PostScript, and that any font, colour, or line thickness changes are local to the embedded PostScript.

To protect against accidental changes to Quikscript procedure or variables, they are all isolated in a separate dictionary, *QSdict*. If you wish to change Quikscript from within embedded PostScript, you should have a PostScript instruction

```
QSdict begin
after %PS%, and
end
```

before PE, as was shown in the "Special page annotation" section above. Alternatively, the other form of embedded PostScript

```
%PS instructions %
```

should be used.

### Embedding Quikscript in a PostScript file

Occasionally, it is better to embed Quikscript and the text within a PostScript file. An example of this is to prepare an advertisement, where the bounds of the area are fixed, there are graphics on some of the borders, and the text content needs to be neatly laid out within it.



PostScript, Java, X-Windows solutions

## Document Preparation Assistant

### Innovative Designer

Qs Consulting is a small company specializing in electronic delivery of documents and graphical software. It handles classified advertising by designing corporate layouts using PostScript, and can deliver advertisements to newspapers and magazines electronically within one day of receipt of advertising content.

A person is sought who can assist in liaising with corporate clients, writing PostScript programs to represent logos and other graphical elements of advertisements that convey corporate style. The person will deal with the client regularly in advertisement preparation. There is also scope for work in Java in preparing WWW advertisements containing animations.

The successful applicant should be competent in writing PostScript, and desirably in Java as well. Social skills in dealing with clients should be of a high level.

An attractive salary package will be negotiated. Applications should be sent to: Qs Consulting, 8 Lavan Pl, Evatt, ACT 2617 (Fax: 6268 8581) by 25th February.

This can be achieved with:

```

%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 190 340

/MM {72 mul 25.4 div} def

/Width 67 MM def           Dimensions of whole area
/Depth 120 MM def

gsave                     Clip to boundary, for safety
0 0 moveto Width 0 lineto Width Depth lineto 0
    Depth lineto closepath clip
newpath 0 setgray

gsave                     Place graphics at top
0.1 setlinewidth
3 Depth 20 sub moveto
. . .
Width Depth 30 sub dup 0 exch moveto lineto
Width Depth lineto 0 Depth lineto closepath
2 setlinewidth stroke
grestore

                                Place graphics at side
0 0 moveto 20 0 lineto 0 Depth 30 sub dup 20 exch
    lineto lineto closepath fill
gsave
/Palatino-Bold findfont
    [0 17 -17 0 0 0] makefont setfont
1 setgray 15 5 moveto
    (PostScript, Java, X-Windows solutions) show
grestore

%include Qs%               Place Quikscript at this point
                                Set margins and currentpoint according to size and graphics
%PM,10,,1%
%PS /RgtCol Width 3 sub store /lGap 1.1 store %
%PS LftCol Depth 58 sub moveto%

                                Now the text to be typeset
%FN,Hel%%SZ,15%%CT%%BD%
Document Preparation Assistant
%SZ,11%L,.5%Innovative Designer%LT%
%SZ,8.1%P,,3,-.7,0%%FJ%
Qs Consulting is a small company specializing
in electronic delivery of
. . .

(Fax: 6268 8581) by 25th February.
                                Finally return to normal PostScript processing

%PS%
cleartomark pop
grestore                   Turn off clipping which was set above
%showpage                 May be wanted

```

## Appendix A: Quikscript Instructions Reference Manual

- %BC,col1,col2,r,g,b%** Set the background colour in a table for the range of columns from *col1* to *col2*. The first column is column 1. *r,g,b* must be in the range from zero to one in value.
- %BC,col1,col2%** Turn off the background colour in the table for the range of columns from *col1* to *col2*.
- %BC%** Turn off the background colour in the table.
- %BD%** **Bold** typeface. To undo this command, use **%LT%**.
- %BG,r,g,b,left,right%** Set the background colour for a block of text. *r,g,b* must be in the range from zero to one in value. The background will be set from the normal left margin minus *left* to the normal right margin plus *right*. These two distances are interpreted as millimeters; if they are omitted, 14 pp and zero respectively will be used.
- %BG%** Turn off the background colour.
- %C,number%** Special characters can be output by specifying the decimal value of the character in the PostScript extended ASCII table. Several numbers can be specified, separated by commas.
- |                         |                   |
|-------------------------|-------------------|
| Eg. Dash — %C,208%      | Bullet • %C,183%  |
| Minus – %C,177%         | Degrees ° %C,202% |
| Minutes ´ %C,194%       | Seconds ¨ %C,205% |
| Pounds £ %C,163%        | Cents ¢ %C,162%   |
| Decimal point · %C,180% |                   |
- The numbers used will change if ISO Latin-1 encoding (European Characters above) is used. See Appendix C for a list of the character codes.
- If you prefer to use octal numeric code as given in the PostScript reference manual and elsewhere, this is possible with **%C,8#number%**.
- %CA,text%** Contents abbreviation. This defines the text which will appear left-abutted to the page number in a table of contents or index. It can be used to specify a short name or number for a chapter. Eg. **%CA,Ch.1-%**
- %CL%** Centre the single line of text that follows between the current left margin and the current column right margin. New-line is not automatically generated (Beware!).
- It is also possible to request that the line be centred about a certain position, with **%CL,pos%**, the position given as MM from the left margin.
- %CO%** Commence a cell in a new column of the table in the current row.
- %CO,n%** Commence a cell in column "n" of the table in the current row, where "n" is a number such as 2, 3, etc.
- %CO,n1,n2%** Commence a cell in column "n1" that will span the columns from "n1" to "n2" of the table in the current row, where "n1" and "n2" are numbers such as 2, 3, etc.

---

%CT%	Centre the following text between the current left and right column margins. Text will still be measured to see how many words will fit on each line, and lines will be broken accordingly unless deliberately broken by some other formatting instruction. Alternatives to this are FJ, LJ, or RJ. See also CL and RL.								
%CW, <i>n</i> %	<p>Column width, or more correctly, position of the right-hand column margin relative to the current left column margin, measured in millimetres. For single-column output this is no different from the page width. For single- or multi-column output, this enables local control of the right margin in the same way that %TB controls the local left margin.</p> <p>Its effect can be undone with %CW%, which reverts to the previous column width. The column width can be increased or decreased by having a "+" or "-" before the number. %CW is useful for building tables.</p> <p>This is unrelated to the "column" of a table which is specified through the %ST and %CO instructions. It refers to the right margin of text within an ordinary document whether the pages are multi-columnar or a single column.</p>								
%DF, <i>n,ch</i> %	Dot fill using <i>n</i> as an optional position on the page, and <i>ch</i> as an optional fill-character. It has a similar effect to %TR, but in moving rightwards to the given position, dots are placed regularly along the current line. If the number is not given, dots fill out to the right margin. If a character or numeric code is given after the number, this character is used instead of a dot.								
%EC%	Stop recording text for an entry in table of contents. Add the page number. See %SC, %PC, %OC, %CA.								
%EH%	Stop recording text for the page header that will appear beside the page number. See %SH.								
%EI%	Stop recording text for an entry in the index. Add the page number. See %SI, %PI, %OI, %CA.								
%ET%	End the table, restoring column margins and justification rule to the state prior to the table. Leave the current point at the lower right corner.								
%FI%	Resume the default behaviour of output line filling (after a %NF directive). The lines of input are split into words, and output so as to fill the lines within the set margins.								
%FJ%	Set full-justification (left and right) for output of following text. Alternatives to this are LJ, RJ, or CT. See also CL and RL.								
%FN, <i>name</i> %	<p>Change font family. The font name is given by a three or four letter code. Unrecognized codes will be ignored.</p> <table border="0" style="margin-left: 40px;"> <tr> <td>Tim (Times)</td> <td>Hel (Helvetica)</td> </tr> <tr> <td>Cou (Courier)</td> <td>Sym (Symbol)</td> </tr> <tr> <td>Pal (Palatino)</td> <td>NewC (New Century Schoolbook)</td> </tr> <tr> <td>Book (Bookman)</td> <td>Ava (Avant Garde)</td> </tr> </table>	Tim (Times)	Hel (Helvetica)	Cou (Courier)	Sym (Symbol)	Pal (Palatino)	NewC (New Century Schoolbook)	Book (Bookman)	Ava (Avant Garde)
Tim (Times)	Hel (Helvetica)								
Cou (Courier)	Sym (Symbol)								
Pal (Palatino)	NewC (New Century Schoolbook)								
Book (Bookman)	Ava (Avant Garde)								

---

	<p>Cha (Zapf Chancery)      HelN (Helvetica Narrow)  Din (Zapf Dingbats)      Uft User-defined</p> <p>%FN, Tim% (Times-Roman) is the default. Only the first four families are guaranteed to be present in a PostScript machine.</p>
%FN%	<p>If no name is given, the font in use before the last change will be used. A history of up to eight fonts is maintained, allowing progressive undo of font changes to a limited extent.</p> <p>Earlier versions of Quikscript used this as a simple toggle between the two most recent fonts. This behaviour can be restored by uncommenting the line <code>/ToggleFont true def</code> near the head of Qs.</p>
%HL%	<p>Draw a horizontal line across the full width of the table below the current row, or at the top if nothing has yet been placed in the table.</p>
%HL,n%	<p>Draw a horizontal line below cell "n" of the current row, or above if the table is still empty.</p>
%HL,n1,n2%	<p>Draw a horizontal line spanning the cells "n1" to "n2" below the current row, or at the top if nothing has yet been placed in the table.</p>
%HLn,...%	<p>Each of the above forms of line instruction can include a line width "n" before the comma. %HL2% will give a double-width line; %HL3,2% will draw a triple-width line below column two.</p>
%IT%	<p>"Italic", or <i>slanted font style</i>. To undo this command, use %RO%.</p>
%L%	<p>Go to the start of a new line. The horizontal position is based on the left margin set by the most recent %TB instruction, plus the paragraph body alignment. If no output has been placed on a page, %L% will not move down the page.</p>
%L,n%	<p>Go to a new line, leaving an extra 'n' blank lines. %L,0% is equivalent to %L%. %L,-1% will go to the start of the current line. Fractional line spacings are allowed.</p>
%LJ%	<p>Left justify the following output text. This is the default behaviour, which is sometimes called "ragged right". Alternatives to this are FJ, RJ, or CT. See also CL and RL.</p>
%LP%	<p>Move to start of the previous line. In general, this could be achieved with %L,-2%. However, if the current position is at the top of the page, this will have undesirable consequences which %LP% avoids.</p>
%LT%	<p>Light typeface, the opposite of bold.</p>
%NC,number%	<p>Change the following page layout to multi-columns. <i>number</i> is the number of columns. If this command is issued part way down a page, the multi-columnar layout commences from that position, so that text above is not overwritten.</p>
%NC%	<p>Go to the top of the next column. In single column layout, this will cause a new page.</p>

---

%NF%	No fill; lines input will not be reformatted to fill the lines output. Each line of input will become a separate line in output. Words are still checked to ensure that they will fit in the output line, and a new line will be forced if words do not fit. This is used frequently for setting poetry, songs, program listings, addresses, and simple tables. See also %FI% and %\%.
%NF, <i>number</i> %	No fill; if input lines will not fit in the space provided, the line will be split at a word boundary, and the line continuation will be indented by <i>number</i> millimeters. Remembered positions can be specified with the "?" notation; see %W below.
%NP%	Start a new page. If no output has been placed on the current page, no action will result.
%NR%	Start a new row of the current table, moving to the first column for following text.
%NR, <i>number</i> %	Start a new row of the current table, moving to the column indicated by the number for following text.
%OC%	Output the text for the table of contents to the screen or standard output. See also %PC.
%OI%	Output the text for the index to the screen or standard output. See also %PI.
%P%	Start a new paragraph. The parameters of the paragraph will be the same as the previous paragraph.
%P, <i>n1,n2,n3,n4</i> %	By default, paragraphs have the body aligned with the left margin; there is no first line indentation; a blank line is left between paragraphs; and a new paragraph will not start within two lines of the bottom of the page, but instead be placed at the top of the next page.
%P, <i>n1,n2,n3,n4</i> %	Start a new paragraph. Set the body alignment <i>n1</i> MM to the right of the current tab margin. Indent the first line relative to the body by <i>n2</i> MM. Set the inter-paragraph gap to one plus <i>n3</i> blank lines; setting <i>n3</i> to 0 will leave a single blank line between paragraphs, and setting <i>n3</i> to -1 will cause new paragraphs to commence on a new line with no intervening blank line. Set the minimum acceptable distance from the bottom of the page for the start of a new paragraph to <i>n4</i> MM; the default <i>n4</i> is 30 MM. If the distance <i>n4</i> is more easily measured as a number of inter-line gaps, this can be done by preceding the number with the character "L".
	Any of the numeric fields can be altered without affecting the others, by simply not providing numbers for the unchanging field. Eg. %P,,,-1% will go to a new paragraph, leave unchanged the body alignment and first line indentation, and set the inter-paragraph gap to zero (1-1) lines.
	Typical uses of these controls are:
%P,,10%	will cause paragraphs to have their first lines indented by 10 MM, but normal body alignment.
%P,20,0%	would be used for including a quotation in a body of text.

---

- The whole quotation would be indented relative to the main text.
- `%P,0,0,0%` would restore paragraph layout to the default.
- `%P,15,-15%` would have a hanging first line. The body is indented 15 MM, but the first line is aligned with the normal margin. This is useful for references, or for drawing attention to the first word of a paragraph.
- If no output has been placed on a page, a new paragraph request will not move the current point down the page.
- See also `%PF` to set paragraph style without commencing a new paragraph.
- `%PC%` Print the table of contents in the document at this point. Each line of the contents will contain the markers `%DF,?%RL%` before the page number. This will give a dotted line from the text to a horizontal position that must already have been set up with a `%W%`. The page number will be right-justified within the column. It may be necessary to use a narrower column (see `%CW`) to improve appearance. It will also be necessary to change the current page number (`%PN`) if this page is to be inserted early in the document.
- `PE` Flag the end of embedded PostScript; the following text will be processed normally by Quikscript. Note the absence of `%` delimiting characters. This is actually a PostScript procedure within Quikscript.
- `%PF,n1,n2,n3,n4%` Modify the attributes of this and following paragraphs, but do not force a new paragraph immediately. The numbers are the same as for the `%P%` command. It is used mostly to modify the next inter-paragraph gap or the body alignment of the remainder of this paragraph, and is particularly useful in tables where the text in each cell is to commence at the top of its area, but the cells are to have different first line indentation or body alignment.
- `%PH,n1,n2,n3,n4%` This is the same as the `%P%` command, except that the minimum distance from the bottom of the page is increased to 2.5 times the normal distance. This makes it suitable before a paragraph heading, where space for the heading and some of the following paragraph must be guaranteed.
- `%PI%` Print the index at this point in the document. Each item will be on a new line. Multi-column (`%NC`) format should be set first to improve layout.
- `%PM,l,r,t,b%` Set the page margins, the distance from the page edges to the text margins. By default, these are 25, 19, 25, 19 MM for left, right, top and bottom. Numbers omitted are left unchanged. Current point is not altered (so a `%VT` may be needed following this at the start of a document).
- `%PN%` Number the pages, starting at page two on the following page.
- `%PN,number%` Number the pages, starting with the given number on the current page. If the number is zero, page numbering will be disabled.

- 
- %PS *commands*%** Execute the PostScript commands that follow %PS (beware strings).  
 Eg. `%PS /l1Gap 1.8 def%%SZ,12%`  
 will cause text to be output double spaced. This means of embedding raw PostScript instructions within a document is suitable if the instructions can be fitted into a single input line. It allows access to the Quikscript procedures and variables. It is unsuitable for changing the content of text strings. (The %SZ command is required in the above example because the line spacing is calculated from the inter-line gap and the text size, and just altering the inter-line gap factor will not directly alter line spacing.)
- %PS%** The following input text will be interpreted as embedded PostScript by the PostScript interpreter. Anything following on the current input line is ignored. To revert to normal text processing by Quikscript, use the PE PostScript command. The Quikscript environment is no longer current, so accidental redefinition of names of Quikscript procedures or variables is prevented. If you need access to them, the Quikscript dictionary must be used, either with
- ```

    QSdict begin...end
  or
    QSdict /name value put
  
```
- %RJ%** Right justify the following output text. Alternatives to this are FJ, LJ, or CT. See also CL and RL.
- %RL%** The remainder of the line, or all characters up to the next positioning Quikscript command on the current input line, are to be right-justified on the output line.
- If the text is to be right-justified to a certain position, this can be done with `%RL,pos%`, the position in MM.
- %RO%** "Roman", or normal vertical font style, the opposite of Italic.
- %SC%** Start recording text to make an item in the table of contents. See %EC, %OC, %PC, %CA.
- %SH%** Start recording text for the page header, to appear beside the page number. See %EH.
- %SI%** Start recording text to make an item in the index. See %EI, %OI, %PI, %CA.
- %SN,*number*%** Set the font height in printer's points, but do not alter the inter-line spacing.
- %SN%** If no number is given, the size in use before the last change will be used. A history of the most recent eight sizes is maintained, allowing progressive undo of size changes to a limited extent.
- Earlier versions of Quikscript used this as a simple toggle between the two most recent sizes. This behaviour can be restored by uncommenting the line `"/ToggleFont true def`" near the head of Qs.

---

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %ST, <i>pos,pos,pos...</i> % | Start a table based on the given positions that define the margins of each column. The position is measured in MM from the normal left margin, or can be remembered positions (see %W% below). Text following this will be placed in the first column. A trailing comma signifies the right margin of the page. See %CO% to change column, %VL% to draw vertical lines separating selected columns, %HL% to draw horizontal lines below a row, %TJ% to set the justification rule for the columns, %NR% to commence a new row, and %ET% to end a table.          |
| %SZ, <i>number</i> %         | Set the font height in printer's points. A printer's point is 1/72 of an inch. Inter-line spacing is 1.17 times the chosen height. Default size is 12 on 14.                                                                                                                                                                                                                                                                                                                                                                                                     |
| %SZ%                         | If no number is given, the size in use before the last change will be used. A history of the most recent eight sizes is maintained, allowing progressive undo of size changes to a limited extent.<br><br>Earlier versions of Quikscript used this as a simple toggle between the two most recent sizes. This behaviour can be restored by uncommenting the line <code>/ToggleFont true def</code> near the head of Qs.                                                                                                                                          |
| %T, <i>n</i> %               | Move horizontally to the position, measured in millimetres, relative to the default left margin, on the current line only. It provides a local "go to".                                                                                                                                                                                                                                                                                                                                                                                                          |
| %T%                          | For details of %T,?%, see %W% below.<br>If no number is given, go to the current left margin, as set in the most recent %TB plus the current paragraph body alignment.                                                                                                                                                                                                                                                                                                                                                                                           |
| %T,+ <i>n</i> %              | Move horizontally to the position, measured in millimeters, relative to the current left margin, on the current line only.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| %TB, <i>number</i> %         | Move to a new left-hand tab margin, measured in millimetres, relative to the default left margin. All following text will be set to this margin.<br><br>Paragraph alignment will be relative to this position, so %TB,10%P,5% will have the paragraph aligned to 15 MM from the normal left margin. %P,5%TB,10% will have the same effect, except that the first line of this paragraph will commence 10 MM from the normal left margin, because the %TB sets the new margin as well as an immediate repositioning.<br><br>For details of %TB,?%, see %W% below. |
| %TB%                         | Revert to the previous left margin, saving the current margin in case a %TB% is again encountered.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| %TB,+ <i>n</i> %             | Move to and set a new left-hand tab margin, measured in millimeters, relative to the current left margin, remembering the old one.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| %TC, <i>r,g,b</i> %          | Set the colour for following text. <i>r,g,b</i> must be in the range from zero to one.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| %TC%                         | Restore the text colour to black.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

---

---

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %TJ, <i>chars</i> %     | Set the justification rule to apply to each column of the current table. The rule is specified as a lower-case character: 'l' for left justified (the default), 'c' for centred, 'r' for right-justified, and 'f' for fully justified (ie both left and right). The characters are not separated by commas. %TJ,ccr% would mean that the first and second columns were to be centred, and the third right-justified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| %TR, <i>n</i> %         | Move horizontally rightwards to the position, measured in millimeters, relative to the default left margin, on the current line only. If the current position is already to the right of this position, do not move left; current position will be unchanged. Remembered positions can be specified with the "?" notation; see %W below.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| %TR,+ <i>n</i> %        | Move horizontally rightwards to the position, measured in millimeters, relative to the current left margin, on the current line only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| %V%                     | Make the following text a subscript. It is not possible for subscripts themselves to have subscripts or superscripts. The end of the subscript is indicated by %=.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| %VL, <i>n1,n2,...</i> % | Record the table boundary positions where vertical lines are to be drawn, eg %VL,1,2,5% will draw vertical lines at the first, second and fifth table margins defined in the most recent %ST instruction. Numbers can be repeated, indicating that those lines are to have double thickness: %VL,1,1,1,5,5,5% will place thick boundaries at the first and fifth margins, but not at intervening positions. Vertical line information is cleared by a %ST instruction, so %VL must be placed after the start of the table. Vertical line placement can be modified within a table.                                                                                                                                                                                                                                                                                                                                                   |
| %VL%                    | Do not draw vertical lines in the table.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| %VM, <i>n</i> %         | <p>Vertical move from the present position, to the start of the line the requested number of millimeters below. If the new position is below the bottom margin, go to a new page and move down the requested distance. It guarantees that an unbroken space of the requested size is left. If the current position is not at the start of a line, it will be moved to the start of the next line before moving down the requested distance.</p> <p>If you know how many lines rather than millimeters are to be left empty, this can be specified with an "L" before the number (%VM,L-2% will move up by 2 lines).</p> <p>%VM can be used to prevent widow lines at the start of a section or paragraph that might fall at the end of a page. Eg. %VM,20%%VM,-20% will go to the start of a line with at least 20 MM clear space below it. If there is not 20 MM at the foot of a page, it will go to the top of the next page.</p> |
| %VT, <i>n</i> %         | <p>Vertical tab: Go to the start of a new line at the given position from the top of the page, measured in millimeters.</p> <p>%VT,0% moves to the start position on the page. This will vary in response to current font size, to prevent large text from going beyond the strict page limits.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

---

It is also possible to move to a line number remembered from a %W command using the "?" notation.

**%W%** Record the X,Y coordinates of the current position for later use. The position is referred to with the character '?' in VT, T, TB, TR, DF, RL, P, PH, ST, CL or NF instructions. Eg. %W% at a point in a document will record the position so that %T,?% subsequently will tab across to the same horizontal position, and %VT,?% will move to the same vertical position. In use with %P, only the body alignment can be set this way.

**%W,number%** Record the X,Y coordinates of the current position, to be identified with the given number. It may be referred to in VT, T, TB, TR, DF, RL, P, PH, ST, CL or NF instructions with '?number'.

Eg. Very simple tables can be set up by defining tab positions, say at 10 MM, 50 MM, and 100 MM. These positions are recorded with

```
%T,10%W,1% %T,50%W,2% %T,100%W,3%
```

The data in the table is then set by referring in turn to the positions as

```
%T,?1% %T,?2% %T,?3%
```

If a column is subsequently found to be too narrow, it can be widened by changing the numbers in the one line where the settings are recorded.

In general, the %ST method of creating a table is greatly superior to simple use of remembered positions. Up to fifty positions can be remembered in this way.

**%\%** Ignore any following text on the line. This enables comments to be included in a document. If "no fill" (%NF) applies, treat the following line as a continuation of the current one.

**%\_%** Output a blank character, and do not allow the space to be the place where a line break occurs. Sometimes, spaces are wanted within a word, eg. a telephone number 268 8111. To ensure that no line break occurs at the gap, it is represented by 268%\_8111. If more than one space is wanted, a number may be given: eg. %\_,3%.

**%^%** Make the following text a superscript. It is not possible for superscripts themselves to have subscripts or superscripts. The end of the superscript is indicated by %=.

**%=%** Return to normal size and position after super- or subscript.

**%%** Output a % symbol. Whatever character is used to mark Quikscript layout instructions (% by default), then repeating the character enables it to be output in ordinary text.

In many situations, a single % can be typed and it will be treated as a single character to be output. However, Quikscript will look at the next character to see if it can be interpreted as a layout command, and if so, all following characters up to the next % will be taken to be part of that command.

Any of the parameters or procedures of Quikscript may be altered, either in the original program, or within a text document using embedded PostScript. They are all held in the *QSdict* dictionary. The ones most eligible for change are:

- lIGap**            The inter-line gap factor, by default set to 1.17. As mentioned in the example in %PS above, changing this number must be followed by a %SZ command in order to take effect.
- PagStr**            The text string output with page numbers. By default, this is (Page ). This is changed by putting text between %SH% and %EH% layout markers, or with the %PS% form of embedded PostScript:  
Eg. %PS%  
      QSdict /PagStr (Technical Report ) put  
      PE
- PagLen**            Length of the piece of paper. 297 MM for A4 paper, but if the output is to be in landscape mode, 210 MM; 280 MM (11 Inch) for US letter, or 217 MM (8.5 Inch) in landscape mode.
- PagWid**            Width of the piece of paper. 210 MM for A4 paper, but if the output is to be in landscape mode, 297 MM; 217 MM (8.5 Inch) for US letter, or 280 MM (11 Inch) if output is in landscape mode. It is used initially to calculate the printable width, establishing the left and right margins. Subsequently, printable width is set within a document with the %PM command.
- ColGap**            Width of the gap between columns in a multi-column output. 5 MM by default.
- Esc**                The character for delimiting layout instructions. In this document we have used %, but any ASCII character could be used, printable or otherwise. Printable characters are usually easier to edit in a text document. However, care is then needed if that character is likely to occur in the natural text and be misinterpreted as part of a layout instruction. Non-printable characters can be used, but are hard to edit unless the editor is able to display them in some way.
- HdrPre**            The font style and size for the page number and associated text. By default, it is defined with:  
/HdrPre (%FN, Tim%%SZ, 12%%RO%%LT%) def
- Eject**            The procedure that is called to go to a new page. The default procedure is as follows:  
/Eject  
  { PrPagNum showpage NCol 1 gt {SetCol} if ToTop } def  
PrPagNum is the procedure called to increment the page number, and output it with a piece of text at the top of the page if the flag NumberIt is true.  
  
If the number of columns is greater than 1, routine SetCol is called to ensure that output will commence in the left-most column. Finally, in procedure ToTop, the starting position is set to the top of the page.

---

## Appendix B Quikscript Files

All ".qs" files described below should follow the Qs file and either precede the document, or be included within the document at the appropriate position.

|              |                                                                                                                                                                                                                                                                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Qs           | The Quikscript program.                                                                                                                                                                                                                                                                                                                                                      |
| stdcodes     | A text file containing the numeric codes and names of all the special characters in standard PostScript fonts.                                                                                                                                                                                                                                                               |
| rotate.qs    | The PostScript program to modify Quikscript to rotate every page so that output is in landscape format rather than portrait format.                                                                                                                                                                                                                                          |
| draft.qs     | The PostScript program to modify Quikscript so that each page will be drawn with the word "DRAFT" added in large shadow letters across the page.                                                                                                                                                                                                                             |
| border.qs    | Example PostScript routine to modify Quikscript, adding a border with rounded corners to each page.                                                                                                                                                                                                                                                                          |
| pdfnt.qs     | The PostScript program to modify the character encodings of the standard PostScript fonts so that western European characters are available using the correct ISO character codes. This differs from the standard ISOLatin1Encoding in that typesetting characters like ellipsis, trademark, florin, dagger, bullet which are not included in the ISO set are also provided. |
| isolatin2.qs | The PostScript program to modify the character encodings of the standard PostScript fonts so that they use an ISO-Latin-2 encoding vector. Composite characters are created from the accent marks and the base characters.                                                                                                                                                   |
| outlft.qs    | The PostScript program to set up a Helvetica font which draws characters in an outline form.                                                                                                                                                                                                                                                                                 |
| smallcap.qs  | The PostScript program to set up a version of the current font with lower case letters drawn as small capitals.                                                                                                                                                                                                                                                              |
| dijkstra.qs  | A cute font imitating a person's handwriting, good for hand-written correspondence that you can type at the keyboard! (Original obtained from the network.)                                                                                                                                                                                                                  |
| altpagm.qs   | The PostScript routine to change the Quikscript behaviour to alternate page numbering from left to right side of page. It is also necessary to include %PN% for page numbers to appear.                                                                                                                                                                                      |
| cntpagm.qs   | The PostScript routine to change the position of page numbers so that they will be centred at the top of the page. It is also necessary to include %PN% for page numbers to appear.                                                                                                                                                                                          |
| pagrange.qs  | Restrict the pages output to those between certain limits specified in the file. The file must be edited each time it is used to set the desired limits.                                                                                                                                                                                                                     |

---

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2col.qs, 3col.qs | Print the document on a landscape page in two or three columns, suitable for producing a simple leaflet.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 2colleaf.qs      | Print the document in two columns suitable for folding as a simple leaflet. The paper will need to be trimmed at the trim marks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 3narrowcols.qs   | Print the document on a portrait page in three narrow columns, using Courier font and No-Fill mode, printing the text exactly as it is in the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 2p.qs, 2pend     | Print the document two pages per sheet, side-by-side. File "2pend" should be concatenated to the end of the document to ensure that nothing is missing from the last page.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| catex.c          | <p>A version of the UNIX <i>cat</i> command that concatenates several files into one and writes the result to standard output. This version examines the content of every file, looking instructions to include other files at that point. It allows macros to be defined, simplifying the preparation of Quikscript documents. It also allows the current date to be generated automatically. It is described more fully below.</p> <p>This is an extremely useful tool with Quikscript since it removes the necessity for including the above PostScript pieces explicitly in every document that uses them. It also enables PostScript drawing instructions for externally prepared diagrams to be held as separate files, rather than being embedded within the document where they will ultimately appear. If the included file is modified, the whole document will automatically reflect the change when next printed.</p> |
| catm.c           | <p>An embellishment of "catex.c" suitable for performing a mail merge on a set of files. As well as including the functionality of "catex.c", it allows a set of fields to be defined, and will make multiple copies of a set of files, substituting the different values for the fields in each copy.</p> <p>The most common use for this is in preparing a letter in which references are made to "fields" such as name, address, appellation, etc, and having a separate file containing the text for the fields for each intended recipient.</p> <p>The first argument when running "catm" is the "fields" file, defining the names of the fields and a series of values for the fields to be used when "catm" repeatedly copies the remaining files to standard output. "catm" is described in more detail below.</p>                                                                                                        |
| notab.c          | A simple filter program to replace tabs with a sensible amount of white space.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|                                                                                   |                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macros.qs                                                                         | Example of how macros can be used with <i>catex</i> or <i>catm</i> . This example shows how to set up a structured document containing sections, dot-lists, and hanging first line lists which can be embedded.                                                                                                                                            |
| QstoHTML.java                                                                     | Quikscript documents can be converted to HTML format if they need to be displayed with a web browser.                                                                                                                                                                                                                                                      |
| QstoHTML.class, EditBuffer.class, OctalString.class                               | The three files comprise the binary version of QstoHTML.java. Execute the first with a Java virtual machine.                                                                                                                                                                                                                                               |
| epstopnm                                                                          | A Unix script for converting an EPS file into a PNM image. This is most useful as a step in handling illustrations when a Quikscript document is to be converted into HTML. It relies on ghostscript being available to do the work.                                                                                                                       |
| epstogif                                                                          | A Unix script for converting an EPS file into a GIF image. It uses epstopnm, and relies on ghostscript and pbmplus or netpbm being installed.                                                                                                                                                                                                              |
| sortIndex.java, avl.java, sortindex.c, avlsort.c                                  | If Quikscript documents output an index to the output screen when run in a PostScript interpreter, the index can be externally sorted with either the C or Java sort program. This is most useful if the document is organized into several separate chapters or sections, where the indexes from each section will need to be merged into a single index. |
| sortIndex.class, AVLnode.class, AVLsortable.class, IndexItem.class, AVLtree.class | The five files comprise the binary version of the Java program to sort an index. Execute <i>sortIndex</i> with a Java virtual machine.                                                                                                                                                                                                                     |
| userdoc.qs                                                                        | This document. To print this document, you should send Qs and this document to a PostScript printer.                                                                                                                                                                                                                                                       |

**catex** (Concatenate and expand files)

This C program requires the names of one or more files as arguments, and will copy each of them in turn to standard output, eg

```
catex file1 file2 >tempfile
```

The single file ("tempfile" in this case) will contain the contents of "file1" and "file2". In the process of copying the files, certain special text strings are scanned for and processed without copying verbatim.

**File inclusion**

If the string `%include name%` is found, the name is interpreted as a file name; reading of the original file is suspended, and the new file is inserted in the output stream.

A file being included will itself be checked in the same way as the original file for occurrences of instructions such as `%include newname%` and such new files will be inserted appropriately. Calls to include files can be nested to any depth.

A file-name is interpreted relative to the file from which it was called, and simple names are interpreted as being in the same directory. Names beginning with `'..'` are interpreted as being in a higher level directory.

**Current date**

A string `%DATE%` within a file will be replaced by the current local date.

**Macro definition and invocation**

Macros can be defined using the form `%DM,name:string:%`, and such strings will not be copied to the output. These set up text substitutions that will apply to the remainder of the document.

Such macros can be invoked with `%M,name%`, so that this text will be replaced by the string in the definition. Invocation is recursive, by which I mean that the replacement text is not simply copied directly to standard output, but instead is processed as the input stream. Within this replacement text may be other macro invocations (`%M,othername%`), new macro definitions, or `%include..` directives. (A macro cannot be redefined within its own body; our simple syntax will not allow colons to appear in the 'string' part of a definition).

A situation where macros might be useful is if we wanted to use two paragraph styles within the one document. Paragraphs have four attributes, the body alignment, the first line indentation, the inter-paragraph gap, and the minimum distance from the bottom of the page for the start of a new paragraph. If we needed to use the style of paragraph with indented first line for most of the document, but in some areas to switch to a hanging first line style for numbered points, we could define two macros for each style:

```
%DM,Num:%P,15,-10,-.5%:%
```

```
%DM,Par:%P,0,5,0%:%
```

Then to commence a normal paragraph, we would use `%M,Par%`, and commence a

numbered paragraph, %M,Num%. The parameters of the two styles would be defined once, and we could switch between the two styles as needed without needing to remember to reset the parameters of the original style at the end of a section.

Similar styles can be set up for section titles, where a little extra space before the title, a change in font size and paragraph style, and the use of bold will be used systematically for all headings throughout the document.

```
%DM,Head:%L%%SZ,16%%PH,0,0,0%%BD%:%  
%DM,Body:%SZ,11%%P,0,0,0%%LT%:%
```

A file is provided (macros.qs) which contains a set of macro definitions that are useful for structured documents, with lists possibly containing sub-lists. In this way, a layout style can be established in the one place, so that measurements of offsets or indentations do not need to be actually used within the document.

**catm** (Concatenate, merge, and expand files)

## Usage:

```
catm [-f fname] [-s char] [-e end-of-page-text] listfile file2 file3 >tempfile
```

or

```
catm [-f fname] [-s char] [-e end-of-page-text] listfile -o file-once -r file3 file4  
>tempfile
```

or

```
catm [-f fname] [-s char] [-e end-of-page-text] listfile -o file-once -r file3 file4  
-o file2-once >tempfile
```

"catm" will read the first file ("*listfile*"), the field value file, and repeatedly copy the remaining files to standard output, performing substitutions based on information read from the field value file. "*listfile*" is the field values file, and will contain the names of "fields" and sets of values for them. The files repeatedly copied will be searched for references to the field names in <> brackets, and such references to a field will be replaced by the current value of this field as determined from the field value file.

Redirection of this output to a file or a pipe is most common.

The "-f" option, which must precede the "*listfile*", indicates that the field-names are defined in a separate file. This allows the "*listfile*" to contain only field values, without needing the field names to be defined at the top with the first set of fields. This file can also contain the "END=" directive described below, further enabling field content to be completely separated from "catm" processing instructions. Note that "END" is a reserved field name.

The "-s" option allows the field separator character to be changed from the Tab character. This only has effect if the field file is organized with multiple field values per line.

The "-e" option allows the value of an "END=" directive to be specified on the command-line. If field names are given as the first line of "*listfile*", then this option removes any need for a separate file containing this directive. If the END text contains blanks, it will need to be given in quotes. The default end text could be given with

```
-e "%NP%%PS%\nQsdict begin Init ToTop end PE\n"
```

While copying the files:

If the text string "%include *file-name*%" is encountered, the file named will be included in the output at that point. If the text string %DATE% is encountered, the current date will be substituted. (Include processing).

If the text string "<*field-name*>" is encountered, then it will be replaced by the current value associated with the name *field-name* found from the first file (*listfile*) in the above examples. (Field processing).

If the text %DM,*name:string*:% is encountered, it will be taken as defining a macro. Wherever %M,*name*% is subsequently found, the text in *string* will be substituted. (Macro processing)

## COMMAND ARGUMENT PROCESSING:

The list of arguments is interpreted as filenames, except for the flags "-o" meaning output once, and "-r" meaning output repeatedly.

The first file is read and interpreted as a set of field definitions that will be applied to other files (the ones to be repeatedly output).

If a "-o" is encountered after the first file, it indicates that the following files are to be output directly with only "include" processing and macro processing being performed.

If a "-r" is encountered after the first file, it indicates that the following files are to be output repeatedly, in turn using each set of field values from the first file. That is, a set of field values is found from the first file. All files following the "-r" are output using this set of field values. Then the next set of field values is read and the process repeated. On each occasion, full "include" and macro processing is performed.

If neither "-o" nor "-r" are present, all files after the first are repeatedly output with field, macro and include processing.

## FIELD PROCESSING

Files for repeated processing are read in turn and copied to standard output.

If a string *<text>* is encountered in the file, where *text* is a field-name defined in the first file or in a "-f" field-name file, the value of the field is substituted for the string.

This process of substituting and copying all the remaining files is repeated for each of the groups of field definitions in the first file.

The simplest way to set up the fields file is to provide on the first line the names of the fields, Tab separated. Subsequent lines will then contain the values of the fields for the next version of the document being generated.

If the "-f" option is used, then the file specified after "-f" will be read seeking the field names, rather than expecting them at the head of the field values file. The field names will be separated by tabs or new-line characters. There may also be an END= directive. The file of fields will contain values for the fields, tab separated, in the same order as the field names.

When a reference to a *<fieldname>* is found in a document to be output, the current replacement text is treated as if it is part of the input document. As this replacement text is output, it is scanned for any macro references, "include" directives, or other field references which are processed as they are encountered.

## INCLUDE PROCESSING

If the string `%include name%` is found at the beginning of a line,

1. the name is interpreted as a file name;
2. reading of the original file is suspended, and
3. the new file is inserted in the output stream.

A file being included can itself contain `%include newname%` instructions, and such new files will be inserted appropriately. Calls to include files can be nested to any depth. (Including a file recursively will cause an infinite loop.)

A file-name is interpreted relative to the file from which it was called. Names beginning with `'..'` are interpreted as being in a higher level directory.

A string `%DATE%` within a file will be replaced by the current local date.

## MACRO PROCESSING

Within a file that is to be copied to output, macros can be defined using the form `%DM,name:string:%,` and such strings will not immediately be copied to the output. These set up text substitutions that will apply to the remainder of the files as they are copied to output.

Such macros can be invoked with `%M,name%`, so that this text will be replaced by the string in the definition. Invocation is recursive, by which I mean that the replacement text is not simply copied directly to standard output, but instead is processed as the input stream. Within this replacement text may be other macro invocations (`%M,othername%`), new macro definitions, or `%include..` directives. (A macro cannot be redefined within its own body; our simple syntax will not allow colons to appear in the 'string' part of a definition).

### EXAMPLE 1:

The simplest way to perform a mail merge is to ignore the "field-names" file and provide a single file containing a single line with the names of the fields followed by lines giving the field values. This file, called "fields" (say), might look like:

```
NAME \t PERS \t ADDR1 \t ADDR2 \t ADDR3
Mr H. Bloggs \t Henry \t 360 Hindmarsh Dr \t Phillip, ACT 2616 \t Australia
Ms R. McManus \t Rosemary \t 19 McWilliam Cres \t Florey, WA 6517 \t
Mr GA Spence \t Mr Spence \t 40 Plenty Ave \t Lower Hutt \t New Zealand
Mrs P Thomas \t Penni \t Computer Science \t ADFA \t Campbell, ACT 2600
```

("\t" means the Tab character.)

The second file (letter) might look like

```
%include ../pscode/letter.hd.qs%
%DATE%

<NAME>
<ADDR1>
<ADDR2>
<ADDR3>

Dear <PERS>

%FI%We are having a reunion of all students and staff who
...
```

The Unix file "letter.hd.qs" will be found in a directory called "pscode" which is a sister directory to the directory that the file "letter" is in.

This letter could now be output with the Unix command

```
catm fields -o Qs -r letter | lpr -Plaser
```

In this case, the output generated by this *catm* command will be:

```
Quikscript file
letter.hd.qs file
today's date

Mr H. Bloggs
360 Hindmarsh Dr
Phillip, ACT 2616
Australia

Dear Henry

%FI%We are having a reunion of all students and staff who
...
%NP%%PS%
QSDict begin Init ToTop end PE
letter.hd.qs file
today's date

Ms R. McManus
19 McWilliam Cres
Florey, WA 6517

Dear Rosemary

%FI%We are having a reunion of all students and staff who
...
%NP%%PS%
QSDict begin Init ToTop end PE
letter.hd.qs file
today's date

Mr GA Spence
40 Plenty Ave
. . .
```

### EXAMPLE 2:

The names of the fields (ie. the first line in the example above) could be given in a separate file, called "fnames".

This letter could now be output with the Unix command

```
catm -f fnames fields -o Qs -r letter | lpr -Plaser
```

### EXAMPLE 3:

In this more challenging example, an invitation is to be sent to many people, some local, and some remote. The letter will contain name, address and personal identification as in the first example, but will also contain some tailoring of the letter content depending on where the invitee lives.

The field values file, "fields3", could be set up as

```
NAME \t PERS \t ADDR1 \t ADDR2 \t ADDR3 \t WHERE
Mr H. Bloggs \t Henry \t 360 Hindmarsh Dr \t Phillip, ACT 2616 \t Australia \t %M,Local%
Ms R. McManus \t Rosemary \t 19 McWilliam Cres \t Florey, WA 6517 \t \t %M,Remote%
```

```
Mr GA Spence \t Mr Spence \t 40 Plenty Ave \t Lower Hutt \t New Zealand \t %M,Remote%
Mrs P Thomas \t Penni \t Dept Computer Science \t ADFA \t Campbell, ACT 2600 \t \t %M,Local%
```

A second file, possibly called *styles*, would be set up to establish style features that were to apply to all of the letters. It might look like:

```
%DM,Remote:%P%If you would like accommodation arranged...:%
%DM,Local:::%
%include dijkstra%
%FN,Uft%
```

This would be included in the output once, after Qs, and before the letters. It defines the text to substitute for the <WHERE> field, and sets up the font to be used in the letters. The text following DM,Remote must all be on the one line. In this case, no alternative text is provided if the person lives locally; their letter will be shorter.

The letter would commence in the same way as the previous example, but would have <WHERE> at the end of a paragraph to pull in the text defined in the second file.

The command to print the letters would then be:

```
catm fields3 -o Qs styles -r letter | lpr -Plaser
```

If the volume of text in Remote was too great to include on one line (or exceeded the input buffer size of 1024 characters), then Remote could be defined to instead include text from another file:

```
%DM,Remote:%include accom.txt%:%
```

#### EXAMPLE 4:

To put addresses on the envelopes, we might want to use label stationery, which has three columns of eight rows of labels per sheet. We could use the same field values file as previously. We do not want to go to a new page after each person, so we would need to change the default end-of-page processing. We could provide a field-names file containing the line

```
END=%L,.6%%P%
```

or we could more simply specify this on the command line with

```
-e %L,.6%%P%
```

Some trials may be needed to get the spacing correct between labels, and a new-paragraph instruction will be useful to force a new column when we are near the bottom of the page.

The printer we are using might allow text to be placed quite close to the top, left and right margins, but it might be limited at the bottom of the page because of the paper-feed mechanism. We would set up a page style that allowed for this:

```
%PM,0,0,5,36%%PS /ColGap 0 store%%NC,3%%NF%%TB,10%%\%
```

This causes page margins to be set up (%PM,0,0,5,36%), remove any gap between columns (%PS /ColGap 0 store%), make three columns per page (%NC,3%), make input lines be simply output without filling the previous line (%NF%), set the left margin 10mm from the left (%TB,10%), and do not go to a new output line when the next input line is read (%\%).

Then the document to be repeated (called "labels") might look like:

```
<NAME>
<ADDR1>
<ADDR2>
<ADDR3>
```

and the command to print the labels could be:

```
catm -e "%L,.6%%P%" fields -o Qs -r labels | lpr -Plaser
```

### EXAMPLE 5:

An alternative format for the fields file could be adopted. There will be one line per field, and the lines for the first letter must define the names to be used in this and in subsequent letters.

The field values file (fields2) might look like

```
NAME=Mr H. Bloggs
PERS=Henry
ADDR1=360 Hindmarsh Dr
ADDR2=Phillip, ACT 2616
ADDR3=Australia
NEXT
Ms R. McManus
Rosemary
19 McWilliam Cres
Florey, WA 6517

Mr GA Spence
Mr Spence
40 Plenty Ave
Lower Hutt
New Zealand
Mrs P Thomas
Penni
Dept Computer Science
ADFA
Campbell, ACT 2600
```

The letter would be the same as in the example above.

The output pages could be generated with Qs layout using the Unix command

```
catm fields2 -o Qs -r letter | lpr -Plaser
```

so that the Quikscript file (Qs) will be copied once at the start, followed by four copies of "letter", each tailored to a name and address given in the "fields2" file.

**macros.qs** (structured documents)

*macros.qs* is an example of how macros can be used with the *catex* or *catm* programs. A set of Quikscript instructions is given a simple synonym, reducing the amount of typing and in this case allowing most features of a document style affecting size or alignment of text to be held in a single file, avoiding repetition of the same information within a document.

The following macros are given:

%M,Title%	Very large, bold, centred line of text at the top of the page.
%M,Section%	Large, bold text, separated from previous text by a larger inter-paragraph gap, and suitable to head a new major section of the document.
%M,Head%	Large, bold text, separated from previous text by a larger inter-paragraph gap, and suitable for a sub-heading which will not interfere with the surrounding paragraph layout.
%M,Body%	Normal text, preceded by a paragraph gap.
%M,Note%	Small, bold text, intended as a heading for "Detail" text, preceded by an inter-paragraph gap.
%M,Detail%	Smaller text, for giving explanatory detail about material given earlier, starting with an inter-paragraph gap.
%M,Small%	Very small text at the current point in the document. This can be undone with %SZ%.
%M,DList%	Start of a list of dot-points, with the first bullet placed.
%M,DP%	Next dot-point in the current list.
%M,DEnd%	End of the list of dot-points.
%M,List%	Start of a list, using a hanging first line style. Use either %T% or %L% according to preference to enlarge on the hanging text. Use %P% to go to the next list item.
%M,LEnd%	End of a list using hanging first line style.
%M,Code%	The following text is to be output using a monospaced font a little smaller than the normal text.
%M,Norm%	Mark the end of "Code", reverting to previous font and size.

- `%M,Asis%`      Treat all white space and new-lines as intended in the output.
- `%M,Fmt%`      Ignore new-line characters, filling the output lines except when layout marks indicate otherwise.
- `%M,1%` to `%M,10%`      Output the number as white within a black circle.
- `%M,mi%`      Minus sign.
- `%M,rarrw%`      Right-pointing arrow.

Tabs are preset to positions every 15 MM, accessible using `%T,?1%` etc. using numbers from 1-4 and then from 15-19.

## QstoHTML (Convert Quikscript documents to HTML)

HTML is a mark-up language similar to Quikscript in many respects, so it should be feasible to convert a Quikscript document to HTML for viewing with a web browser. QstoHTML does this, quite well for most documents. However, the two layout languages are different in a number of ways that makes some situations difficult to handle properly. Most of the problem areas are dealt with by putting out a comment in the HTML and usually also to the screen, allowing them to be fixed by hand. QstoHTML does a good job with tables, and handles different paragraph styles as best it can, given the limited mechanisms in HTML for representing style.

### Usage

QstoHTML is written as a Java application. It can be compiled (if necessary) with

```
javac QstoHTML.java
```

It can be run with the command:

```
java QstoHTML [-s sub-file] [-c char-file] [-sz numb]  
[input-file [output-file]]
```

This executes the program in files QstoHTML.class, EditBuffer.class, and OctalString.class that are generated by the compilation.

All options should be given on the one command-line. Input comes from *<input-file>* or from standard-input. Output is written to standard output or to *<output-file>*.

Messages are sent to standard error, indicating the line number in the input file where difficulties were encountered in conversion.

### Keyword substitutions

QstoHTML has a large set of fixed conversions, some of which require look-ahead, and some which require the state of the machine to be remembered. However, it is possible to override any of the conversions by reading from a given file the substitutions that are to apply. Keyword substitution is performed when a keyword has been identified, after the leading and trailing "%" character has been stripped off, and before it is checked against the rules embedded in QstoHTML. This allows us to modify any of the default processing performed by QstoHTML if we wish. It also allows us to indicate replacements for Quikscript macros that can be defined for expansion by "catex". A file of substitutions

```
-s sub-file
```

can be provided containing lines with the format

```
string1=string2
```

where *<string1>* is the text between "%" Quikscript delimiters, and *<string2>* is the text to be put in the HTML output file. Eg, if a Quikscript macro was defined:

```
%DM,1:%FN,Din%%C,182%%FN%:%
```

enabling the Dingbats character 1 in a black circle **①** to be represented by %M,1%, then this could be handled in QstoHTML with a line in the substitution file

```
M,1=(1)
```

"(1)" is as good as we can achieve with the limited typesetting features of standard HTML.

## Character substitutions

A number of character encodings are commonly used in PostScript, and these can easily be set up in Quikscript by including PostScript code. Unfortunately, the HTML character set is impoverished, and there are many PostScript characters that cannot be represented in HTML. QstoHTML allows the character encoding to be established through a file

```
"-c char-file"
```

where the corresponding HTML character (or string) is identified. The format of this file is

```
number=repl
```

where <number> is the numeric representation of a character, and <repl> is the text that will be output in the HTML file if that character is encountered. Character substitution is only performed on the actual text of the document after keyword processing has been performed, and in %C,number% Quikscript layout marks. Eg.

```
162=&cent;  
163=&pound;  
177=-
```

If no character substitution file is explicitly mentioned, the file "pschcode" will be read if it is available.

## Default size

Quikscript documents contain explicit size specifications for text, measured in printers points. HTML does not use this scheme, but rather a base-font and sizes relative to this. It is possible with QstoHTML to indicate which PostScript font size is to be taken as the base-font size

```
-sz numb
```

The size must be an integer, and the default size is 13. The number does not need to be an actual size used in the document. Sizes encountered will be compared with this number to determine approximately how much larger or smaller than the default they are. Choosing a default size that is too large will cause small text in PostScript to be almost unreadable on the screen.

## Illustrations

The most common form of illustration used in Quikscript documents is an EPS file. QstoHTML will flag any included PostScript in its output. It is convenient to convert such EPS files to a form displayable in a browser. Two Unix scripts are provided to assist with this. The first is "epstopnm", which uses ghostscript to render the PostScript and create a PPM file. The second is "epstogif", which calls "epstopnm" and then "ppmtogif", a pbmplus (or netpbm) utility.

## Problem areas

Quikscript allows any PostScript to be embedded within it, which can have a wide variety of effects on layout and representation of following text. QstoHTML simply outputs the PostScript code as a comment in the HTML, and also puts a message to the screen showing the first three lines of the PostScript to warn that it has not been converted.

The purpose of many Quikscript layout marks is to precisely position text. HTML has no concept of precise positioning. It does not even support tabs. QstoHTML ignores many layout marks, and performs a rough substitution in other cases.

HTML does not allow all text styles to accumulate, so that `<I>` for italic and `<B>` for bold are mutually exclusive rather than additive. It has a scheme for treating every character as intentional, similar to `%NF%` in Quikscript, but it only renders such text in monospaced font.

HTML has the notion of embedding of one style within another, and most browsers require that the termination of styles be done in the reverse sequence of their declaration. So,

```
<I>test<FONT SIZE=-1>smaller</FONT></I>
```

may not be rendered properly if the styles overlap, as in

```
<I>test<FONT SIZE=-1>smaller</I></FONT>
```

Quikscript does not have this concept of embedding, so layout marks in generated HTML may not always be in the correct order. Some hand-editing of the HTML may be required to overcome such problems. Any desired hot links will also have to be added by hand.

## **epstopnm** (Convert EPS files to PNM images)

EPS files can be converted to image files using ghostscript. PNM format is a portable format allowing easy conversion to other formats such as GIF or TIFF, which are used in much other software. It is particularly useful in converting PostScript files included in Quikscript documents into GIF format for display within a web browser.

This Unix shell script is based on a script *pstopnm* written by Alberto Accomazzi and distributed with ghostscript. It differs from the original in the way the size of the image is determined, and in the command arguments that can optionally be used.

The script is run with the command:

```
epstopnm [-llx s] [-lly s] [-urx s] [-ury s] (printers points)
          [-xsize n] [-ysize n] (size in pixels)
          [-pbm|-pgm|-ppm]
          [-rot]
          [-forceplain]
          [-verbose] [-help]
          psfile[.e]ps]
```

where the optional arguments would all appear on the same command line.

### **Input area size**

The page area rendered into the image is determined from the BoundingBox details within the PostScript file, which must be present. This area can be modified by specifying all of the `-llx`, `-lly`, `-urx`, `-ury` optional arguments. A 4-pp margin is always added automatically.

### **Image size**

The size of the output image can be specified by giving one or both of `-xsize` and `-ysize`. Otherwise its pixel size will be 100/72 times the rendered area size in printers points.

### **Output format**

Output will go to a file or files based on the name of the input file. Names are generated of the form *psfile*001.ppm, *psfile*002.ppm. The format of this file will be "ppm" by default, suitable for colour images, but "pbm" for black-and-white or "pgm" for gray-scale images can alternatively be chosen. If the `-forceplain` option is specified, the data is written in a raw format, binary data rather than ASCII, giving a more compact output file.

The image can be rotated by 90 degrees with the `-rot` option.

## **epstogif** (Convert EPS files to GIF images)

This Unix script uses *epstopnm*, so it relies on ghostscript being installed. It also uses *ppmtogif*, a *pbmplus* (or *netpbm*) utility program. It will convert an EPS file directly into a GIF image. It takes command-line arguments which it passes to *epstopnm*. It creates the file *psfile*.gif.

**sortIndex** (sort an index containing text and page numbers)

This program is provided in three forms, as C source code, and as Java source code and binary, any of which can be used.

C code: compile with

```
gcc sortindex.c avlsort.c -o sortindex
```

and run with

```
sortindex < inputfile > outputfile
```

Java code: compile with

```
javac avl.java sortIndex.java
```

and run with

```
java sortIndex < inputfile > outputfile
```

index.

Quikscript has two instructions for dealing with an index, %PI% to print an index, and %OI% to output the index to standard output, which will be the screen if running in a PostScript interpreter such as ghostscript, or could be a message-logging file if run in a PostScript printer. %PI% is appropriate to use if the whole document can be processed in one run through Quikscript. %OI% is used if the document is organized into several files that are impractical to handle in a single pass through Quikscript, or if the sorting used in Quikscript is inadequate.

The *sortIndex* program is intended for use with the output from %OI% commands. It reads from standard input, and writes to standard output the sorted index. The index files from different sections should first be concatenated into a single file, in the order of the chapters or sections in the final document. If an index item is found in several places, *sortIndex* will combine them into a single index item, with the page information from both pages in the order in which they appeared in the input to the program.

Example: If I have a document consisting of three chapters, with each chapter in a separate file.

- Within the first chapter, I might indicate that page numbers are to commence with "1-", the second chapter with "2-", etc. This is achieved with %CA,1-% in the first chapter and so on.
- At the end of the each chapter file is the instruction %OI%. If this file is run with Quikscript in a PostScript interpreter such as ghostscript, the window from which ghostscript is run will have the index text displayed in it.
- I will then use the mouse to select this text, paste it into an editor, and save it.
- The other chapters would be set up and processed similarly.
- Then the three indexes should be combined into a single file.
- If the first two chapters both contained an index item %SI%fractal compression%EI%, then the combined index file should contain a line similar to

```
fractal compression 1-7
```

and the line

```
fractal compression 2-4
```

It is important that the index from the first chapter be placed before the index from the second chapter in the combined index file.

- When *sortIndex* is run on this file, it will be sorted on the basis of the text before the page number. Case of the text is ignored in sorting, so upper- and lower-case will be interspersed rather than having all upper-case preceding all lower-case as will be done in sorting within Quikscript.

The two items will be combined into a single index item:

```
fractal compression 1-7, 2-4
```

- The output from *sortIndex* would then need to be given appropriate Quikscript heading text, and printed using Quikscript. This header might look like:

<code>%PM, 25, 25, 25, 25%</code>	<i>Set suitable page margins</i>
<code>%SH%Index-%EH%%PN%</code>	<i>Number the pages clearly</i>
<code>%SZ, 20%%BD%%CL%Index</code>	<i>Give a centred heading to the page</i>
<code>%LT%%SZ%%P%</code>	<i>Revert to original font, and new paragraph</i>
<code>%NC, 3%%LP%%NF, 7%</code>	<i>Set 3 columns, with no-fill processing starting next line</i>

## Appendix C Other Useful Software

### ghostscript

A PostScript previewer is an invaluable tool to assist in developing Quikscript documents. "ghostscript" or its related versions "gsview" and "ghostview", is particularly recommended, and it will run on most platforms. Versions of it are freely available from many sites, including

<http://www.cs.wisc.edu/~ghost/>

To preview a Quikscript document with ghostscript, copy Quikscript and your document into a single file, and type

```
gs yourfilename
```

As well as checking layout, ghostscript allows non-PostScript printers to be used for printing PostScript documents and Quikscript files. There are drivers included for most types of printer.

Ghostscript has (or will soon have) a component for converting a PostScript file into a PDF (portable document format) file. PDF files are binary, and are much smaller than PostScript. They are intended for viewing with a PDF viewer, such as AcroRead, which is freely available from Adobe Systems (<http://www.adobe.com/prodindex/acrobat>), and which can be easily set up to be called directly by a web browser if a fetched document is in PDF format.

The ghostscript tool *ps2pdf* will create a PDF file. It is structured as separate pages which can be accessed in any order. This means for the Quikscript user that there is an easy mechanism to convert a Quikscript document into a form viewable by a web browser. However, it is not possible to include hot links within the document as it is in binary format.

There is also a tool *pdf2ps* which will generate a PostScript file from a PDF file. Combined use of these two tools gives a way of converting a Quikscript document into one structured with distinct pages but still in PostScript format. This can be a useful way to prepare lecture material.

Another tool in ghostscript is *ps2epsi* which produces a document in EPS format, suitable for including within another PostScript document, with bounding-box dimensions at the top, and other comments giving an image representation of the document which can be viewed in some document preparation systems.

**distillery**

Multi-page documents can easily be created using Quikscript. However, some page viewing software works better if the pages are distinct, with no dependence between them, and printable in any order. This is impossible to achieve with Quikscript alone, because it reads the document as an input stream and dynamically determines where the line- and page-breaks should occur.

"Distillery" is a program that converts almost any PostScript program into separate distinct pages conforming to the PostScript Document Structuring Conventions. The original version of this program was developed by Glenn Reid when he worked for Adobe Systems Inc. The most recent version of this program is maintained by Graham Freeman and is available from <ftp.adfa.edu.au> in directory *pub/postscript*. See <http://www.cs.adfa.edu.au/~gfreeman>

The simplest way to use this program is to set up the PostScript file to be distilled, for instance by concatenating Quikscript with the document. Distillery can be run in a PostScript previewer such as *ghostscript* with:

```
gs
(distillery) run
(mypsfile) distill
quit
```

This will create a file "mypsfile" (the original file name with an 'x' added). This file will no longer contain Quikscript; it will have a short header and then raw PostScript drawing instructions. Some fonts and some embedded graphics produced with external packages do not survive the distillation process accurately.

**Qse**

Qse is a companion program to Qs. It is written in Java, and should be portable to most platforms. It allows use of menus for previewing and printing documents, and gives ready access to the other ancillary files for font or style. It includes a set of fonts that work on PostScript machines.

Although Qs can be used without Qse and for many people this would be its preferred method of use, Qse provides a pleasant environment in which to create and work with Quikscript documents.

## Appendix D Special Characters in PostScript Fonts

Almost all fonts contain the following characters, accessible with their numeric codes:

161	ı	162	ç	163	£	164	/	165	¥	166	f	167	§	168	œ
169	'	170	“	171	«	172	<	173	>	174	fi	175	fl	177	–
178	†	179	‡	180	·	182	¶	183	•	184	,	185	„	186	”
187	»	188	...	189	‰	191	¿	193	`	194	´	195	^	196	~
197	-	198	˘	199	·	200	¨	202	°	203	¸	205	ˆ	206	˙
207	˘	208	—	225	Æ	227	ª	232	Ł	233	Ø	234	Œ	235	°
241	æ	245	ı	248	ı	249	ø	250	œ	251	ß				

If the program "pdfnt.qs" is run, the special characters will change to the following:

28	ˆ	30	°	128	•	129	†	130	‡	131	...	132	—	133	–
134	f	135	/	136	<	137	>	138	–	139	‰	140	„	141	“
142	”	143	‘	144	’	145	,	146	™	147	fi	148	fl	149	Ł
150	Œ	151	Š	152	Ÿ	153	Ž	154	ı	155	ı	156	œ	157	š
158	ž	161	ı	162	ç	163	£	164	œ	165	¥	166	ı	167	§
168	¨	169	©	170	ª	171	«	172	¬	174	®	175	-	176	°
177	±	178	²	179	³	180	´	181	µ	182	¶	183	·	184	¸
185	¹	186	°	187	»	188	¼	189	½	190	¾	191	¿	192	À
193	Á	194	Â	195	Ã	196	Ä	197	Å	198	Æ	199	Ç	200	È
201	É	202	Ê	203	Ë	204	Ì	205	Í	206	Î	207	Ï	208	Ð
209	Ñ	210	Ò	211	Ó	212	Ô	213	Õ	214	Ö	215	×	216	Ø
217	Ù	218	Ú	219	Û	220	Ü	221	Ý	222	Þ	223	ß	224	à
225	á	226	â	227	ã	228	ä	229	å	230	æ	231	ç	232	è
233	é	234	ê	235	ë	236	ì	237	í	238	î	239	ï	240	ð
241	ñ	242	ò	243	ó	244	ô	245	õ	246	ö	247	÷	248	ø
249	ù	250	ú	251	û	252	ü	253	ý	254	þ	255	ÿ		

For eastern European languages, the file "isolatin2.qs" will give these special characters:

161	À	162	˘	163	Ł	164	œ	165	Ł	166	Š	167	§	168	¨
169	Š	170	Ş	171	Ť	172	Ž	173	-	174	Ž	175	Ž	176	°
177	ą	178	˙	179	ı	180	´	181	ı’	182	ś	183	^	184	¸
185	š	186	ş	187	ı’	188	ž	189	ˆ	190	ž	191	ž	192	Ř
193	Á	194	Â	195	Ã	196	Ä	197	Ł	198	Č	199	Ç	200	Č
201	É	202	Ě	203	Ë	204	Ë	205	Í	206	Î	207	Ď	208	Đ
209	Ň	210	Ň	211	Ó	212	Ô	213	Õ	214	Ö	215	×	216	Ř
217	Û	218	Ú	219	Ů	220	Ü	221	Ý	222	Ť	223	ß	224	ı
225	á	226	â	227	ă	228	ä	229	Í	230	ć	231	ç	232	č
233	é	234	ę	235	ë	236	ě	237	ı	238	î	239	đ’	240	đ
241	ń	242	ň	243	ó	244	ô	245	õ	246	ö	247	÷	248	ř
249	ů	250	ú	251	ů	252	ü	253	ý	254	ı	255	˙		

The Symbol font contains the following characters:

a	α	b	β	c	χ	d	δ	e	ε	f	φ	g	γ	h	η
i	ι	j	φ	k	κ	l	λ	m	μ	n	ν	o	ο	p	π
q	θ	r	ρ	s	σ	t	τ	u	υ	v	ϖ	w	ω	x	ξ
y	ψ	z	ζ												
A	Α	B	Β	C	Χ	D	Δ	E	Ε	F	Φ	G	Γ	H	Η
I	Ι	J	ϑ	K	Κ	L	Λ	M	Μ	N	Ν	O	Ο	P	Π
Q	Θ	R	Ρ	S	Σ	T	Τ	U	Υ	V	ς	W	Ω	X	Ξ
Y	Ψ	Z	Ζ												
0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
8	8	9	9	!	!	"	∇	#	#	\$	∃	%	%	&	&
'	ε	(	(	)	)	*	*	+	+	-	-	/	/	,	,
.	.	:	:	;	;	<	<	=	=	>	>	?	?	@	≡
[	[	\	∴	]	]	^	⊥	-	-	'	>	{	{		
}	}	~	~												
161	Υ	162	'	163	≤	164	/	165	∞	166	f	167	♣	168	♦
169	♥	170	♠	171	↔	172	←	173	↑	174	→	175	↓	176	°
177	±	178	"	179	≥	180	×	181	∞	182	∂	183	•	184	÷
185	≠	186	≡	187	≈	188	...	189		190	—	191	↙	192	⌘
193	ℑ	194	℔	195	⊗	196	⊗	197	⊕	198	∅	199	∩	200	∪
201	⊃	202	⊇	203	⊄	204	⊂	205	⊆	206	∈	207	∉	208	∠
209	∇	210	®	211	©	212	™	213	∏	214	√	215	.	216	¬
217	^	218	∨	219	↔	220	⇐	221	↑↑	222	⇒	223	↓↓	224	◇
225	⟨	226	®	227	©	228	™	229	Σ	230	(	231		232	\
233		234		235		236		237	{	238		239		240	
241	⟩	242	↓	243	∩	244		245	∪	246	)	247		248	∩
249		250		251		252		253		254		255			

Most machines also provide the ZapfDingbats font:

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	✱												
0	✂	1	✂	2	✂	3	✓	4	✓	5	✕	6	✕	7	✕
8	✕	9	+	!	✂	"	✂	#	✂	\$	✂	%	✂	&	✂
'	✂	(	✂	)	✂	*	✂	+	✂	-	✂	/	✂	,	✂
.	✂	:	+	;	+	<	+	=	+	>	+	?	+	@	+
[	✱	\	✱	]	✱	^	✱	_	✱	'	✱	{	'		'
}	“	~	”												
161	☪	162	☪	163	☪	164	♥	165	☪	166	☪	167	☪	168	♣
169	◆	170	♥	171	♠	172	①	173	②	174	③	175	④	176	⑤
177	⑥	178	⑦	179	⑧	180	⑨	181	⑩	182	①	183	②	184	③
185	④	186	⑤	187	⑥	188	⑦	189	⑧	190	⑨	191	⑩	192	①
193	②	194	③	195	④	196	⑤	197	⑥	198	⑦	199	⑧	200	⑨
201	⑩	202	①	203	②	204	③	205	④	206	⑤	207	⑥	208	⑦
209	⑧	210	⑨	211	⑩	212	➔	213	➔	214	↔	215	↕	216	➤
217	➔	218	➤	219	➔	220	➔	221	➔	222	➔	223	➔	224	➔
225	➔	226	➤	227	➤	228	➤	229	➤	230	➤	231	➤	232	➤
233	➤	234	➤	235	➤	236	➤	237	➤	238	➤	239	➤	240	
241	➤	242	➤	243	➤	244	➤	245	➤	246	➤	247	➤	248	➤
249	➤	250	➤	251	➤	252	➤	253	➤	254	➤	255			

## Index

- % in text 6
- % 6, 35, 51
- %=% 19, 51
- %\% 12, 51
- %^% 19, 51
- %\_ 19, 51
- %BC 27, 43
- %BD% 6, 43
- %BG 26, 43
- %C, 18
- %C,% 43
- %CA% 29, 30
- %CL% 12, 43
- %CO% 43
- %CT 20, 44
- %CW 18, 44
- %DF, 22, 44
- %DM 56
- %EC% 29, 44
- %EH% 44
- %EI% 30, 44
- %ET% 22, 44
- %FI% 11, 44
- %FJ% 20, 44
- %FN 44
- %FN% 13
- %HL 22, 45
- %include 54, 56
- %IT% 6, 45
- %L 5, 45
- %LJ% 20, 45
- %LP% 45
- %LT% 6, 45
- %M 56, 64
- %NC 17, 45
- %NF 11, 46
- %NP% 5, 46
- %NR 46
- %OC 46
- %OI 46
- %OI% 70
- %P 14, 46
- %PC% 47
- %PF, 47
- %PH 17, 47
- %PI% 47, 70
- %PM, 18, 47
- %PN 19, 47
- %PS 48
- %RJ 20, 48
- %RO% 6, 48
- %SC% 29, 48
- %SH% 48
- %SI% 30, 48
- %SN 12, 48
- %SN,% 13
- %ST, 22, 49
- %SZ, 12, 49
- %T, 7, 49
- %TB, 7, 49
- %TC 26
- %TC, 49
- %TJ, 24, 50
- %TR, 50
- %V% 19, 50
- %VL, 22, 50
- %VM, 16, 50
- %VT, 16, 50
- %W 51
- /Esc 35
- ? 10
- Advertisement example 41
- Alignment of text 20
- Alternating page number 53
- avlsort 55
- Background 26
- Background colour 27, 43
- Block indentation 7
- Bold 6, 43
- Bold deselecting 6
- Border drawing 53
- Border to page 35
- Catex 54, 56
- Catm 54, 58
- Centred line 12, 43
- Centred page number 53
- Centred text 20, 44
- Changing font 13, 44
- Character codes 36, 53
- Characters, special 18
- Code listing 11, 46
- ColGap 52
- Colour 26, 43, 49
- Column colour 43
- Column of table 43
- Column width 18, 44
- Columns, number 17, 45
- Comments within document 12, 51
- Contents 29, 44, 46, 47, 48
- Currentpoint 39
- Czech characters 53, 74
- Dash 18
- Dijkstra font 53
- Distillery 73
- Document Structuring
  - Conventions 73
- Dot Fill 44
- Dotted line 22
- Double spacing 33
- Draft 36, 53
- Eject 52
- Em-dash 18
- Embedded graphics 39
- Embedded PostScript 33, 47, 48
- Embedded PostScript,
  - guidelines 40
- Embedding Quikscript 41
- End contents item 44
- End index item 44
- End of header 44
- End of table 44
- Envelope addresses 62
- EPS files 40
- Epstogif 55, 69
- Epstopnm 55, 69
- Esc 52
- European characters 36, 53, 74
- Files with Quikscript 53
- Filling lines 11, 44
- Font change undo 13
- Font family 38
- Font for page header 52
- Font name 13, 44
- Font size 12, 48, 49
- Font size undo 12
- Full justification 44
- Gap between columns 52
- Ghostscript 72

- 
- Graphics embedded 39
  - Gs 72
  - Hanging first line 15
  - HdrPre 52
  - Header 48
  - Header text 44
  - Heading placement 17
  - Headings 12
  - Horizontal line 39, 45
  - Horizontal tab 7
  - HTML 55, 66
  - Ignored text 12
  - IIgap 52
  - Including graphics 39
  - Indentation of block 7
  - Index 30, 44, 47, 48
  - index sorting 70
  - Inter-line spacing 12, 33, 52
  - Inter-paragraph spacing 15
  - ISO encoding 74
  - ISO-Latin1 encoding 53, 74
  - ISO-Latin2 encoding 53, 74
  - Italic 6, 45
  - Italic deselecting 6
  - Justification of text 20
  - Keyword character 52
  - Landscape 35, 53
  - Layout marker character 35
  - Layout markers summary 43
  - Leading 12
  - Leaflets 54
  - Left justify 45
  - Left margin 16, 49
  - Length of page 52
  - Light font 45
  - Line breaks respected 11
  - Line previous 45
  - Macros 55, 64
  - Mail merge 54, 58
  - Margins 18
  - Mathematics 19
  - Minus 18
  - Multi-column layout 17
  - New line 5, 45
  - New page 5, 46
  - New page procedure 52
  - New row 46
  - No fill 46
  - Non-breaking space 19, 51
  - Not filling lines 11
  - Notab 54
  - Number of columns 17, 45
  - Numbered points 9, 16
  - Outline font 37, 53
  - Output contents 46
  - Output index 46
  - Page border 35
  - Page header 26
  - Page heading 52
  - Page margins 18, 47
  - Page numbering 19, 33, 34, 47
  - Page range 53
  - PagLen 52
  - PagStr 52
  - PagWid 52
  - Paragraph 5, 46
  - Paragraph format 47
  - Paragraph gap for heading 17
  - Paragraph style, numbered 9
  - Paragraph styles 14
  - PDF 72
  - PDF font 53
  - Pdf2ps 72
  - PE 47
  - Poetry 11
  - Polish characters 53, 74
  - Position remembering 10
  - PostScript embedded 33, 47, 48
  - Preformatted text 11, 46
  - Previous line 45
  - Print contents 47
  - Print index 47
  - Ps2epsi 72
  - Ps2pdf 72
  - Qse 73
  - QstoHTML 55, 66
  - Quikscript editor 73
  - Range of pages 53
  - Remembering position 10, 51
  - Right justify 48
  - Roman 48
  - Rotate 53
  - Running title 33
  - Size of font 12
  - Small capitals font 37, 53
  - sortIndex 55, 70
  - Space, non-breaking 19
  - Spacing between lines 33
  - Special character 18, 43
  - Special characters 36, 74
  - Stack 39
  - Start contents item 48
  - Start header text 48
  - Start index item 48
  - Start table 49
  - Subscript 19, 50
  - Summary of layout markers 43
  - Superscript 19, 51
  - Symbol font 75
  - Tab 49
  - TAB removal 54
  - Tab rightwards 50
  - Table 22, 49
  - Table background 27
  - Table justification 50
  - Tabulation 7
  - Tabulation of a block 7
  - Tabulation of block, toggling 8
  - Text colour 26, 49
  - Three-column leaflet 54
  - Two pages per sheet 54
  - Two-column leaflet 54
  - Vertical line 50
  - Vertical movement 50
  - Vertical placement 16, 50
  - Where 51
  - Width of page 52
  - ZapfDingbats 76
-

## Contents

Lines, Paragraphs .....	5
Bold, Italic .....	6
Horizontal tab .....	7
Numbered Points .....	9
Remembering position .....	10
Not Filling Lines (new-line taken seriously) .....	11
Headings, Font Size, Centring .....	12
Changing font .....	13
Paragraph styles .....	14
Vertical Tab .....	16
Headings at Page Bottom .....	17
Multi-column layout .....	17
Page margins .....	18
Column width .....	18
Special character .....	18
Non-breaking space .....	19
Page numbering .....	19
Superscripts, Subscripts, Mathematics .....	19
Full Justification, Centred, Right Justification .....	20
Dot Fill .....	22
Table .....	22
Text colour .....	26
Background Colour .....	26
Table background colour .....	27
Table of Contents .....	29
Index .....	30
Embedded PostScript .....	33
Double Spacing .....	33
Special page annotation and numbering .....	33
Landscape Orientation .....	35
Layout marker character .....	35
Graphic Page Border .....	35
European/ISO Characters .....	36
Outline Font .....	37
Small Capitals Font .....	37
Embedded Graphics .....	39
External Illustration .....	39
Embedding Quikscript .....	41
Appendix A: Instructions Summary .....	43
Appendix B: Quikscript Files .....	53
catex .....	56
catm .....	58
macros.qs .....	64
QstoHTML .....	66
epstopnm .....	69
epstogif .....	69
sortIndex .....	70

Appendix C: Other useful software .....	72
ghostscript .....	72
distillery .....	73
Qse .....	73
Appendix D: Special Characters .....	74