UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 67

November 1986

Title:

A SCRIBE REFERENCE MANUAL

Author:

R.J.Hare, C.D.McArthur, A.McKendrick

Contact:

Advisory service

Software Support Category:
See Note 15.

Synopsis

This document describes the facilities available in version 3.6 of the local implementation of the SCRIBE document preparation program. This implementation is available on all ERCC controlled mainframe machines running a dialect of the EMAS operating system, and is designed to aid the preparation and production of various types of document. Its command structure is compatible with that of the similarly named documentation suite marketed by Unilogic (available locally on the ERCC VAX/VMS machine).

The document is effectively a continuation of, and should be read in conjunction with the locally produced SCRIBE primer [5] which serves to introduce the user to the basic philosophy and mode of operation of SCRIBE and which describes briefly the use of SCRIBE at a simple level. The examples in this document are intended to represent SCRIBE manuscript input exactly as it might appear in the SCRIBE manuscript file.

It should be noted that even taken together, the locally produced primer and manual are intended to be no more than a statement of what facilities are available within the local implementation of SCRIBE and should not be treated as a comprehensive guide to the use of SCRIBE. This function is fulfilled more than adequately by the Unilogic documentation [8] which describes in full the use of the facilities described here and in the primer, and which is **essential** reading for any user intending to use SCRIBE for anything other than the simplest of jobs. Reference copies of this document are available for inspection in the ERCC Advisory offices at 59 George Square and the James Clerk Maxwell Building, and copies may be purchased at cost price from the same locations.

Document prepared using SCRIBE and printed on a Xerox X2700 laser printer
Diagrams prepared using NOTICE.

Keywords

Document Formatting, SCRIBE, Text Formatting

Edinburgh Regional Computing Centre

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

Table of Contents

1 Introduction

This document describes the facilities available within the SCRIBE text processing system available on ICL 2900 series machines (EMAS and BUSH), and the Amdahl machine (EMAS-A) at ERCC. The document is effectively a continuation of the locally produced SCRIBE 'primer' [5], and users should be familiar with the contents of that document before proceeding to read this document. In particular, instructions for setting up and running SCRIBE on EMAS, BUSH and EMAS-A are contained in the primer, which also contains a brief description of SCRIBE, its underlying design philosophy and the adoption of its command format by ERCC.

Not all of Unilogic SCRIBE's functionality has been implemented locally, and facilities described in the Unilogic manual [8] may not be available in the local implementation, so this User Note is to be regarded as the definitive statement of what is available on the EMAS, BUSH and EMAS-A implementations of SCRIBE.

Users requiring a more detailed description of SCRIBE's facilities than is contained here or in the primer must consult the Unilogic manual [8], reference copies of which are available for inspection in the ERCC Advisory offices at 59 George Square and the James Clerk Maxwell Building. Copies of this manual may be purchased at cost price from the same locations.

The article by Browning [1] is a brief overview of some of the problems associated with computerised document preparation, and a statement of 'what is required' from document preparation software, though it is directed more towards the micro/mini computer solution, and SCRIBE is not discussed. The review article by Furuta et al. [7] discusses SCRIBE in the context of several other text-processing programs and will give interested users a deeper insight into the design philosophy of modern text processing systems than is contained either here, or in the Unilogic manual.

2 SCRIBE's Language and Basic Terminology

In order that no confusion arise, the convention adopted in this document is that EMAS commands are referred to as 'commands', and SCRIBE commands are referred to as 'instructions'. SCRIBE instructions are the formatting directives entered into the input (manuscript) file along with the text to be processed. This is the file which will ultimately be processed by SCRIBE to produce the output (document) file.

SCRIBE instructions will generally adhere to one of the following formats:

@instruction
@instruction(text)
@instruction(argument)
@instruction(argument,keyword$_1$/value$_1$,..,keyword$_n$/value$_n$)
@instruction(argument$_1$=argument$_2$,keyword$_1$=value$_1$,..,keyword$_n$/value$_n$)
@instruction(keyword$_1$ value$_1$,..,keyword$_n$ value$_n$)

All **instructions** are prefixed by the @ character and may be followed by a delimited list which may comprise; some **text** on which the instruction operates; an **argument** which may be a named environment, and perhaps a list of **keyword value** pairs all enclosed within bracketing delimiters (in the above cases, the delimiters are brackets; (and)). Items within the list are separated from one another with commas (,).

For example the instruction, `@DEVICE(LPT)` instructs SCRIBE to prepare the current manuscript for printing using a standard line printer as the output device.

Note that there must be no space between the @ character and the instruction since the string "@ⱥ" (ie: @ followed by a space) has a special meaning of its own (see section 4).

SCRIBE allows any of the following matched pairs of delimiters:

[and], (and), { and }, < and >, ' and ', " and "

For example, @MAKE"REPORT" is equivalent to @MAKE(REPORT). In this document, the different pairs of delimiters are extensively used for illustrative purposes.

Upper and lower case can be mixed in all SCRIBE instructions.

For example, @DeViCe(LpT) is exactly equivalent to @dEvIcE(LpT). Note that in this document, all SCRIBE instructions, keywords etc. are shown in upper case for the sake of clarity.

text, argument and **keyword value** pairs are separated from one another with commas (,).

The **keyword value** pairs are used to further modify or control either an instruction or the overall document style. As indicated in the examples above, they may be separated by either a space ( ), an equals sign (=) or a slash (/) – in this document a space is usually used. **keyword** specifies some named attribute or property of an environment or instruction, and **value** specifies the value of that attribute. In the event that **value** specifies some parameter which has units, if the units are specifed, either of the forms shown below may be used:

```
SPACING 3INCH
SPACING 3 INCH
```

Allowable units of distance for use with **keyword/value** pairs are; IN, INCH, INCHES; CM, CENTIMETRES; MM, MILLIMETRES; PT, POINT, POINTS; PICA, PICAS; EM, EMS; QUAD, QUADS; CHAR, CHARS, CHARACTER, CHARACTERS; EN, ENS; LINE and LINES. The default unit for vertical distances is LINE or LINES, and for horizontal distances; CHAR, CHARS, CHARACTER or CHARACTERS.

SCRIBE's basic function is to format text 'objects' such as letters, words, sentences, paragraphs, etc., into a pleasing printed layout, for example by indenting paragraphs or by separating sentences with more than one space. SCRIBE has to be able to recognise these objects and to do so applies the following basic rules to the manuscript text:

- 'words' must be separated by spaces or 'end of line'
- 'sentences' must be terminated by a full stop character, (ie; ".", "?" or "!") and have at least 2 spaces or a 'newline' after them
- 'paragraphs' must be separated by at least one blank line (ie: 2 adjacent 'newline' characters)

In addition, there will be at any time a large number of further rules in effect, governing for example:

- what to do with multiple spaces
- what to do with multiple blank lines
- what the margins should be
- which font to use
- what if anything to underline
- what the vertical spacing should be, etc.

Such a set of rules defines an **environment** which has a name. For each rule or **attribute** that can be specified there is a corresponding environment keyword-value pair: for example `BLANKLINES KEPT` or `JUSTIFICATION OFF`.

There are many predefined text environments (see sections 5 and 8) which can be used as they stand or with modifications (see sections 3.7 and 6.3), and new environments can be defined easily (see section 3.7), by assigning new attribute values with the appropriate **keyword=value** pairs.

SCRIBE recognises several types of document and holds a database of environments for each of these document types.

SCRIBE instructions themselves fall into two general classes:

- instructions which produce an immediate one-off effect at the place they appear, for example "take a new page" or "tab to the next tab stop"
- instructions which cause an environment change, thus affecting an arbitrary number of factors for an arbitrary duration.

3 SCRIBE Instructions

SCRIBE recognises the following instructions, which have been gathered together by similarity of function as far as possible.

Any Unilogic SCRIBE instruction or facility which is **not** described in the remainder of this document may be assumed not to have been implemented in the local version of SCRIBE.

3.1 Basic Instructions

`@BEGIN`<**environment**>	specifies that subsequent text is to be interpreted in the context of a new environment, eg: `@BEGIN[DESCRIPTION]` Attributes of the 'old' environment are not lost when a new environment is entered. Environments may be nested and the new environment may inherit some or even most of the old one's attributes if they have not been respecified. Details of the environments which are generally available are given in section 5. Some document types (eg: `LETTER`) allow the use of special environments within that document type only, these are described under the relevant document type in section 8.
`@COMMENT`{**comment text**}	specifies text in the manuscript which SCRIBE is to ignore in producing the output file, eg:

`@COMMENT{Comments do not appear in SCRIBE output}`

`@DEVICE`"**device name**"
specifies the device type for which SCRIBE has to prepare the document file, eg:

`@DEVICE{GP300}`

Further details of the device types known to SCRIBE are given in section 9.

`@END[`**environment**`]`
specifies the end of an excursion into the named environment, eg:

`@END"VERSE"`

`@FOOT{`**text**`}`
text will be inserted in the processed document either as a footnote, an endnote, or a note 'inline', according to the value of the `@STYLE` keyword `NOTES` (see section 6.2), eg:

`..@FOOT<See Brown pp 141-147.>..`

`@INCLUDE(`**file**`)`
specifies that the text of **file** is to be processed exactly as if it had been encountered in the manuscript file at that point. **file** may contain text, SCRIBE instructions (except `@DEVICE` and `@MAKE` instructions), or both, eg:

`@INCLUDE(ERCY02.NOTEFORMAT)`

This instruction is particularly useful for setting up (say) departmental document formats in a file which is made available to all departmental users via the EMAS `PERMIT` command (see the relevant EMAS User's Guide [2], [3]), thus the example above could well illustrate the use of a standard departmental file for defining a format for lecture notes.
The included file may be a member of a partitioned file (and additionally, on EMAS-A a group member), in which case it must be specified as such. Note that a maximum of 8 files may be incorporated in a document file in this way.

`@MAKE<`**document type**`>`
specifies which of the document types held in SCRIBE's internal database is to be used for this manuscript, eg:

`@MAKE{MANUAL}`

Details of all document types known to SCRIBE are given in section 8.

`@MESSAGE{`**message text**`}`
specifies text to be displayed at the user's terminal.

3.2 Spacing Control Instructions

`@BLANKPAGE`"**n**"
inserts **n** blank pages into the document file once the current output page has been filled - it does not leave a partially filled page as `@NEWPAGE` (see below) does, eg:

`@BLANKPAGE{2}`

`@BLANKSPACE'`**vertical distance**`'`
inserts blank space of the specified depth in the document file at this point (to leave space for a small figure perhaps), eg:

`@BLANKSPACE'4 INCHES'`

Units may be any suitable units from the list given in section 2.

@HINGE — marks a point at which text enclosed in the GROUP environment (see section 5.7) **may** be broken, if such text extends over more than one page, eg:

```
@HINGE
```

@HSP{**horizontal distance**} — causes blank space of the specified width to be placed in the document file, eg:

```
@HSP<1.75 INCH>
```

@NEED[**vertical distance**] — if less than **vertical distance** is left on the current page, skips to the next page, eg:

```
@NEED(12)
@NEED{4.6 INCHES}
```

The NEED instruction is for use with 'blocks' of text of known length which are required to be entirely on the same page on the output (document) file.[1] Note that the @NEED instruction and the NEED keyword (see section 6.6 are **not** the same.

@NEWPAGE[**n**] — causes SCRIBE to skip immediately to the start of the next page, and then leave **n** blank pages before resuming the output of processed text.

3.3 Tab Control Instructions

Tab control instructions are used in conjunction with the single-character tab instructions (see section 4.2) to set up and create tables. As tables are intrinsically unfilled text 'objects' it follows that tabs are (almost) always set up and used in unfilled environments.

@TABCLEAR — clears all existing tab stop settings.

@TABDIVIDE(**n**) — divides the available formatting area into **n** equal width columns. Thus:

```
@TABDIVIDE[6]
```

sets 5 tab stops each 1/6th of the way across the available width, with the right margin being treated as the final tab stop.

@TABSET<**stop1,..,stopn**> — sets a series of **n** tab stops at the specified horizontal positions and does not erase existing tab settings. Horizontal distances are calculated with respect to the currently prevailing left margin, not relative to the edge of the paper, eg:

```
@TABSET(1.0 INCH, 2 INCHES, 3.5 INCHES)
```

3.4 Page Footing and Heading Instructions

The text printed at the top (or bottom) of a page is called a page heading (or footing). The default page heading is a page number centred at the top of every page after the first, but this format may be changed if required. The heading and footing areas are

[1]The @NEED instruction is not available in Unilogic SCRIBE.

each subdivided into three parts, a left part, a centre part and a right part, and control of these six areas is provided through the following two instructions.

`@PAGEFOOTING(LEFT` **"text"**, `CENTRE` **"text"**, `RIGHT` **"text"**, `ODD/EVEN)`
allows you to specify a footing to be placed at the bottom of a page. Only those keywords required need be specified. ODD or EVEN allows different footings to be specified for alternating pages in a double-sided document. An equals sign (=) or a slash (/) may be used in place of a space to separate the field from its value, eg:

```
@PAGEFOOTING{RIGHT "@VALUE<PAGE>", ODD}
@PAGEFOOTING{LEFT= "@VALUE(PAGE)", EVEN}
```

`@PAGEHEADING` takes the same keyword parameters as above but of course is used to specify the format of a page heading to be placed at the top of every page, eg:

```
@PAGEHEADING{CENTRE "DRAFT", LEFT "SCRIBE
 Reference Manual", EVEN}
```

The default page heading is a centred page number. This will be automatically 'turned off' the first time a @PAGEHEADING instruction is used, so care must be taken to set up a new page number field if this is required (in either a @PAGEHEADING or @PAGEFOOTING instruction).

3.5 Sectioning Instructions

SCRIBE allows the insertion of sectioning instructions into the appropriate document types. SCRIBE will then automatically keep a count of these sections and will make use of them when preparing a table of contents. The arguments to the sectioning instructions are used as section titles when the table of contents is generated.

It is important to appreciate that titles generated using the sectioning instructions differ from those generated using the heading environments (see section 5.6 below) in the sense that they appear in the table of contents while the titles generated by the heading environments do not. The instructions are:

`@APPENDIX(`**title**`)`	produces a section title for an Appendix, eg:
	`@APPENDIX(Use of EMAS for FORTRAN Programmers)`
`@APPENDIXSECTION<`**title**`>`	produces an Appendix section title, eg:
	`@APPENDIXSECTION"Logging In"`
`@CHAPTER'`**title**`'`	produces a Chapter title, eg:
	`@CHAPTER<Online Help Information>`
`@PREFACESECTION"`**title**`"`	produces an unnumbered title for a preface section, eg:
	`@PREFACESECTION<Preamble>`
`@PARAGRAPH[`**title**`]`	produces a paragraph title, eg:
	`@PARAGRAPH{A Paragraph Title}`
`@SECTION<`**title**`>`	produces a section title, eg:
	`@SECTION[Standard Environments]`
`@SUBSECTION[`**title**`]`	produces a subsection title, eg:

```
@SUBSECTION(Information on Current SCRIBE Run)
```

`@UNNUMBERED{`**title**`}` produces an unnumbered section title, eg:

```
@UNNUMBERED(References and Further Reading)
```

The sectioning instructions are used by SCRIBE to set up a section numbering system when used in document types which allow this. These section numbers are used by SCRIBE when generating a table of contents in documents which allow this. The table below summarises the numbering hierarchy used by SCRIBE when this is done.

Textual Level	ARTICLE Instructions	REPORT and MANUAL Instructions	Normal Numbering Format
1	@PREFACESECTION	@PREFACESECTION	unnumbered
1	@SECTION	@CHAPTER	n.
2	@SUBSECTION	@SECTION	n.n.
3	@PARAGRAPH	@SUBSECTION	n.n.n.
4	-	@PARAGRAPH	n.n.n.n.
1	@UNNUMBERED	@UNNUMBERED	unnumbered
1	@APPENDIX	@APPENDIX	x.
2	@APPENDIXSECTION	@APPENDIXSECTION	x.n.

3.6 String and Counter Handling Instructions

`@SET(`**counter value**`)` sets the value of **counter** to **value**. Allowable counters are page and section numbers, (see section 3.5 above). Values may be absolute, incremental or decremental. An equals sign (=) or a slash (/ may be used to separate **counter** from **value**, eg:

```
@SET(PAGE 29)
@SET<CHAPTER=+7>
@SET[SECTION=-4]
```

`@STRING<`**name**`="`**value**`">` provides a method of setting up frequently used long text strings, by defining a name for the string which is given within quotes, eg:

```
@STRING(ERCC = "Edinburgh Regional Computing
 Centre")
```

The current content of a string can be recalled using the @VALUE instruction described below.

`@VALUE(`**name**`)` causes the value of the specified string **name** to be inserted in the document file at this point, eg:

```
@VALUE(ERCC)
```

In addition to strings defined by the user with @STRING, the @VALUE instruction may also access the values of certain predefined strings. These are described in section 7.

3.7 Instructions for Changing Things

`@COUNTER<`**name, keyword value**`,..>`
allows the definition of new counters, eg:

```
@COUNTER[SUBPARAGRAPH, WITHIN PARAGRAPH,
 NUMBERED "@£@:.@1", REFERENCED "@£@:.@1",
TITLEFORM "@PARM(NUMBERED)"]
```

The @PARM instruction is used for the transfer of 'parameter' values within SCRIBE, and its use is fully described in the Unilogic documentation. @COUNTER is an 'advanced' instruction, and its use should not be undertaken without consulting the Unilogic documentation. [8] [9],

@DEFINE<**environment, keyword value,..**>
allows new environments to be defined by specifying an environment name and a list of keyword-value pairs which describe the layout of the environment, eg:

```
@DEFINE(RAGGEDTEXT, FILL, JUSTIFICATION OFF,
 SPACING 1)
```

For a fuller description of the @DEFINE instruction and a description of its associated keywords and their values, please see sections 6.5 and 6.6.

@DEFINEFONT<**fontname, facecode="typeface",..**>
on those devices with multiple typefaces available, allows the definition of 'fonts' which consist of a collection of facecode name->typeface name assignments. Such assignments may well differ from those contained in the 'standard' font, eg:

```
@DEFINEFONT[MYFONT, I="COMPMODITALIC11",

R="COMPMODROMAN11"]
```

Currently only one font is defined for each SCRIBE device, and great care should be exercised in defining fonts independently. In order that this be done successfully, a full understanding of the way SCRIBE operates in this area is essential, and this will only be obtained by consulting the Unilogic manual [8].

@EQUATE(**newenvironment=oldenvironment**)
allows 'aliases' to be given to any other SCRIBE environments, for example in the case where a long environment name is to be used many times, or perhaps when a different spelling is desired, eg:

```
@EQUATE{PE=PROGRAMEXAMPLE}
@EQUATE[ITEMISE=ITEMIZE]
```

The @EQUATE instruction may also be used to give aliases to instructions, eg:

```
@EQUATE<BP=BLANKPAGE>
```

@FORM
A powerful form of the @TEXTFORM instruction (see below). This advanced instruction is fully described in the Unilogic manuals [8], [9], and its use should not be undertaken without consulting the said documents.

@MODIFY[**environment, keyword value,..**]
allows the modification of an existing environment or counter, eg:

```
@MODIFY(TEXT, SPACING 2)
```

For full details of the @MODIFY instruction, and its associated **keyword=value** pairs when usid to modify an

environment, please see sections 6.3 and 6.6. Modification of counters is outwith the scope of this document - those interested should consult the Unilogic Database Administrator's Guide [9].
Onlythose excursions into an environment, or uses of a counter occurring after a @MODIFY instruction will be affected by the changes specified in that @MODIFY instruction.

@STYLE[**keyword value,..**] specifies (usually) a global style for the whole document. The STYLE setting will affect the appearance of the whole document, not just individual environments (which are effected with @MODIFY), eg:

```
@STYLE(JUSTIFICATION OFF, SPREAD 2)
```

For a full description of the STYLE instruction and its associated keywords please see sections 6.1 and 6.2.

@TEXTFORM[**name=**"...@PARM(**text**)..."]
allows the insertion of a variable text string in a fixed 'template' which may consist of text, SCRIBE instructions or both. The text string is 'inserted' into the template via the @PARM component of the instruction, eg:

```
@TEXTFORM[NAME="My name is @PARM[TEXT]."]
```

defines a textform called NAME which is used as follows:

```
..@NAME[Roger Hare]..
```

For a full explanation of the use of the @TEXTFORM and @PARM instructions; also for instructions in the use of the more flexible @FORM instruction, and the @PARMQUOTE instruction, consult the Unilogic documentation [8], [9]. These are 'advanced' instructions, and their use should not be undertaken without consulting the documents referred to above.

3.8 Indexing Instructions

SCRIBE has sophisticated facilities for indexing. There are five indexing instructions which allow for:

- simple indexing, with a topic keyed to a page number.
- as above with a bold page number to indicate an entry of major significance.
- secondary indexing; that is, a major topic with a subsidiary list of topics included under the main heading.
- as above with a bold page number to indicate an entry of major significance.
- tertiary indexing whereby a related (indexed) topic is pointed to with an entry of the "see also topicname" type entered under a selected major topic (if secondary indexing is also included under that major topic, the "see also" reference is placed after the secondary index).

Indexing is achieved merely by inserting the desired indexing instruction at the appropriate point in the manuscript. The index is generated automatically in document types which allow an index, and is sorted on an ASCII based key with upper and lower case letters being equivalenced.

`@INDEX<`**text**`>` simple primary indexing of the entry **text**, sorted alphabetically, eg:

```
@INDEX<INDEX instruction>
```

`@INDEXMAIN(`**text**`)` as above, but with a boldface page number, eg:

```
@INDEXMAIN{Indexmain instruction}
```

`@INDEXSECONDARY[PRIMARY="`**primaryentry**`", SECONDARY="`**secondaryentry**`"]`

secondary indexing of **secondaryentry** under **primaryentry**. **primaryentry** is sorted alphabetically in the context of all the other primary index entries, and **secondaryentry** is sorted alphabetically in the context of all secondary entries under the relevant primary entry, eg:

```
@INDEXSECONDARY[PRIMARY="Instructions",
 SECONDARY="Indexsecondary instruction"]
```

`@INDEXSECONDARYMAIN{PRIMARY="`**primaryentry**`", SECONDARY="`**secondaryentry**`"}`

as above but with a boldface page number, eg:

```
@INDEXSECONDARYMAIN[PRIMARY="Instructions",
 SECONDARY="Indexsecondarymain instruction"]
```

`@SEEALSO{PRIMARY="`**primaryentry**`", OTHER="`**otherentry**`"}`

inserts a subsidiary entry of the "see also **otherentry**" form subordinate to the top level index entry **primaryentry**, eg:

```
@SEEALSO{PRIMARY="Modify instruction",
 OTHER="Define instruction"}
```

The index to this document has been constructed using the `@INDEX` and `@INDEXSECONDARY` instructions.

3.9 Cross Referencing Instructions

SCRIBE allows you to refer to other parts of your document by means of four cross referencing instructions:

`@LABEL<`**codeword**`>` marks a selected point in the text which may be referred to with the `@PAGEREF` and `@REF` instructions. **codeword** is the string which identifies the labelled point, eg:

```
@LABEL{CROSSREFERENCINGCOMMANDS}
```

`@LABEL` is used to label parts of the text which are going to be referred to on a section number or page number basis.

`@PAGEREF"`**codeword**`"` is used to refer forwards or backwards to a LABELled or TAGged item in the text. **codeword** is the label part of the relevant `@TAG` instruction. The reference is to the page number on which the item is found in the processed document, eg:

```
@PAGEREF(CROSSREFERENCINGCOMMANDS)
```

`@REF(`**codeword**`)` is used to insert a forward or backward reference to a previously TAGged or LABELled point in the manuscript file. **codeword** is the label inserted at the appropriate point of the manuscript file, eg:

```
@REF{CROSSREFERENCINGCOMMANDS}
@REF{TAGPARAGRAPH}
```

	The reference is to the current section number, so the document type being used must be one of the sectioned document types (see section 8.3), and, the sectioning instructions must be in use.
`@TAG[`**codeword**`]`	is used to label parts of the text which are to be referred to on a 'text object' basis, eg: the third item in an `ITEMIZE`d list. **codeword** is the label used by the `@REF` instruction to refer to the appropriate part of the text, eg: `@TAG"TAGPARAGRAPH"`

4 Single Character Instructions

Many of the non-alphanumeric characters, used singly, are also defined as SCRIBE instructions, mainly for use with tab control. They are gathered together in this section for convenience, but also should be considered along with the relevant tab control instructions described in section 3.3.

4.1 Format Control

`@*`	when used in filled text, forces SCRIBE to begin a new line immediately. The broken line is not justified.
`@|`	marks a position within a word where a line break may be taken, if necessary. This instruction may be used to mark hyphens in hyphenated words, for example. Note that in such circumstances, the hyphens must be inserted by the user - this is **not** an automatic hyphenation facility.

4.2 Table Format Control

`@^`	sets a tab stop at the current position.
`@\`	is the tab instruction for moving the notional "cursor" (printing position) to the next tab stop. It also marks the right hand end of the text being "centred" "flushed right" or replicated (see below).
`@=`	marks the left hand end (beginning) of text to be centred between tab stops. It should not be used in a "filled" environment.
`@>`	marks the left hand end (beginning) of text to be flushed right against the next tab stop. It should not be used in a "filled" environment. Before flushing text against the extreme right hand margin of the line, a tab stop must be set at the extreme right hand end of the line. Failure to do this will result in a `line too long` message (see page 38) being displayed.
`@$`	sets the left margin for the current environment to the current position on the line.
`@!`	sets the return marker to the current position on the line.
`@/`	moves the cursor to the return marker position.
`@&`	causes the characters between the `&` and the next tab instruction in the manuscript to be replicated (ie: duplicated as often as is necessary) in the output from the current cursor position up until the next tab stop.
`@)`	is similar to `@&` but the replicated pattern is positioned such that all such filling patterns given subsequently will line up underneath each other.

These instructions are used largely with the tab control instructions (see section 3.3) to control the layout of unfilled tables. Please consult the Unilogic User Manual [8] for full details.

4.3 Miscellaneous Single Character Instructions

`@@`	generates the `@` character itself in the document.

5 Standard Environments

This section describes the predefined standard environments. They can be specified in one of two forms normally depending on the length of text to be formatted. The short form of an environment instruction has the general form:

`@`**environment**`(text to be formatted)`

The long form is bracketed by the `@BEGIN` and `@END` instructions:

```
@BEGIN(environment)
     .
text to be formatted
     .
@END(environment)
```

The long form allows modifications to be made to the environment, just for this one use of it; eg:

`@BEGIN(`**environment**`, SPACING 3)`

would enter the new environment with `SPACING 3` overruling the default spacing for **environment**. For full details of the environment keywords see section 6.6.

One environment may be embedded inside another one.

There are two main classes of formatting environments, referred to as "filled environments" and "unfilled environments".

In a filled environment, SCRIBE pays no attention to the line breaks in the manuscript file, and fills each line in the document file with words from the manuscript file until that line is full. Note that the filling of lines should not be confused with justification (where the words are positioned on lines such that the ends of the lines are all aligned). However, in practice the SCRIBE database is set up such that all filled environments will be justified unless this is overridden (by use of the `@STYLE` or `@MODIFY` instructions).

In an unfilled environment, line breaks are significant and SCRIBE does not fill lines in the document file by adding more words until the output line is full. Each line in the manuscript file produces one equivalent line in the document file (although this is not necessarily true with blank lines). It should be noted that if a line in the manuscript file exceeds the specified line length for the document file, an error message is produced, but no other action is taken.

5.1 Filled Environments

`TEXT`	is the default environment for running text but it is occasionally necessary to specify it explicitly to get text inside (say) an unfilled environment. For example, the bulk of this document is set in the `TEXT` environment.
`QUOTATION`	highlights prose quotations in running text by insetting the text with wider margins and vertical spacing, eg:

```
@BEGIN<QUOTATION>
Speak not about what you have read but
about what you have understood.
@END<QUOTATION>
```

VERSE — is similar to QUOTATION except that a new line is started in the document file for each line break in the manuscript file. Long lines are not truncated but are wrapped round and indented slightly more than the prevailing left margin. Thus the VERSE environment would be more accurately labelled as "semi-filled", eg:

```
@VERSE{Give me a spirit that on this life's rough sea
Loves t'have his sails filled with a lusty wind
Even till his sail-yards tremble, his masts crack,
And his rapt ship runs on her side so low
That she drinks water and her keel plows air.}
```

5.2 Unfilled Environments

FORMAT — reproduces the contents of the manuscript file exactly as encountered. A variable-width font is used (if the output device type supports it). This means that for tabular output it is essential to use tab control instructions within the body of the text - otherwise the columns won't in general line up, eg:

```
@FORMAT'FORMAT prints text
exactly as encountered
in the manuscript file
using a variable
width typeface.'
```

VERBATIM — is rather like FORMAT, the only difference being that a fixed width typeface is used. VERBATIM is ideal for the reproduction of short tables and does not require the use of tabbing controls in such circumstances, eg:

```
@BEGIN(VERBATIM)
This text is printed
exactly as encountered
in a fixed width typeface.
@END(VERBATIM)
```

A variant of the VERBATIM environment is PROGRAMEXAMPLE which uses a fixed width typeface other than the default one. This is useful for highlighting examples of computer input and output.

DISPLAY — is similar to VERBATIM except that SCRIBE indents both left and right margins. It also separates the displayed text from the document body using vertical spacing, eg:

```
@BEGIN[DISPLAY]
This is a bit like
VERBATIM except that
the margins are widened.
@END[DISPLAY]
```

EXAMPLE — is similar to DISPLAY except that it uses a different typeface to highlight the text. It is particularly useful for setting off examples from surrounding descriptive material, eg:

```
@EXAMPLE(This is like
DISPLAY except that
a different typeface
is used.)
```

5.3 Itemising Environments

The itemising environments are all filled.

DESCRIPTION highlights paragraphs with a header word at the left margin. Subsequently lines of the paragraph are indented significantly to cause the header word to stand out. Where necessary the tab instruction (@\) should be used to separate the heading word from the rest of the text (see section 3.3). The descriptions of most of the SCRIBE instructions described in this document are set using the DESCRIPTION environment.

ENUMERATE produces paragraphs which are numbered (outer level 1,2,3 etc. nested inner level a,b,c etc.), eg:

```
@BEGIN(ENUMERATE)
first item

second item

last item
@END[ENUMERATE]
```

ENUMERATEd lists may be nested to as deep a level as required, but the style of numbering will start to repeat at the fourth level.

ITEMIZE is rather like ENUMERATE, except that instead of being numbered, paragraphs are flagged with a special character (such as an asterisk (*) or dash (-)). The paragraphs are set off from the rest of the text by vertical spacing and wider margins, eg:

```
@BEGIN<ITEMIZE>
first item

second item

last item
@END<ITEMIZE>
```

ITEMIZEd lists may be nested to as deep a level as required, but the style of itemization will start to repeat at the third level.

5.4 Positioning Environments

The positioning environments are all unfilled.

CENTRE[2] requests that each line in the body of the environment be centred between the global margins, eg:

```
@CENTRE(SCRIBE USERS GUIDE)
```

FLUSHLEFT forces the first character in each of the manuscript lines to be

[2]The spelling CENTER is used in Unilogic SCRIBE.

aligned with the global left margin, eg:

```
@BEGIN(FLUSHLEFT)
This text is to be flushed left.
@END(FLUSHLEFT)
```

`FLUSHRIGHT` forces the last character in each manuscript line to be aligned with the global right margin, eg:

```
@FLUSHRIGHT<This text is to be flushed right.>
```

5.5 Facecode Environments

The following paragraphs describe the predefined facecode environments. These instructions select a new 'typeface', thus affecting the appearance of the processed text at the level of individual characters. In particular, the spacing of characters will generally be affected. This is one of the situations where SCRIBE may have to give the nearest equivalent to what is requested if the selected output device does not support the typeface requested; eg: a line printer would use the normal (and only!) typeface when Greek characters were requested.

The instructions are shown without an @ prefix as they may be used in either the @**facecode** or @BEGIN(**facecode**)......@END(**facecode**) form.

`B` requests **boldface** printing (this is sometimes achieved by repetitive overprinting on primitive devices) eg:

```
@B{boldface}
```

`C` requests SMALL CAPITAL printing, eg:

```
@C'small capital'
```

`G` specifies that the bracketed text is to be printed in γρεεκ letters, eg:

```
@G[greek]
```

`R` requests that printing revert to the ordinary Roman typeface which is the default. This is required to produce ordinary letters inside one of the other facecode environments.

`T` requests that the bracketed text be output in a `typewriter` font, eg:

```
@T(typewriter)
```

`I` requests *italic* printing, eg:

```
@I"italic"
```

SCRIBE will use underlining on devices that do not support italics.

`P` combines bold and italic, requesting ***boldface italics***, eg:

```
@P(boldface italics)
```

`U` requests that each "non-blank" character in the bracketed text be underlined, eg:

```
@U{This text will be underlined}
```

`UN` requests that only letters and digits within the bracketed text be underlined, eg:

```
@UN<Only letters and numbers are underlined here>
```

`UX` requests that all characters in the bracketed text be underlined, eg:

```
@UX"Here, everything will be underlined"
```

`+` causes the enclosed text to be printed as a $^{\text{super}}$script at the current position, eg:

`@+(super)script`

- causes the enclosed text to be printed as a $_{\text{sub}}$script at the current position, eg:

`@-{sub}script`

5.6 Heading Environments

HEADING	produces a normal heading which is centred on a line and printed where possible in medium-sized letters. It is used in simple non-sectioned document types, eg: `@HEADING{This is a heading}`
MAJORHEADING	produces a major page heading, printed in large letters (where possible) and positioned in the centre of the line. It is used for simple non-sectioned document types and is neither numbered nor entered in a table of contents, eg: `@MAJORHEADING<This is a major heading>`
SUBHEADING	produces a subheading flush to the left margin. It is printed in normal sized letters and underlined. As with the above two environments, it is not numbered, eg: `@SUBHEADING[This is a subheading]`

5.7 Miscellaneous Environments

GROUP The GROUP environment ensures that the enclosed text appears on the same page and that therefore a new page must be started if there is not enough space remaining on the current page for the enclosed text, eg:

```
@BEGIN[GROUP]
  .
  .
@END[GROUP]
```

W tells SCRIBE to treat the enclosed text as one word, eg: This is very useful in sequences where there are significant blank spaces which should not be split over two lines or be 'padded out' with extra space when the text is justified , such as personal names, eg:

`@W"R.J.Hare, C.D.McArthur, A.McKendrick"`

6 Changing Things – @MODIFY, @DEFINE and @STYLE

The environment attributes (or properties) described briefly in section 2 above may altered by specifying them in the appropriate instructions as follows:

@instruction{..,keyword value,..}

where **keyword** and **value** are respectively the name of the attribute to be changed, and **value** is its value. **keyword value** pairs may be separated with a space, an equals sign (=) or a slash (/). In this document, the space is usually used.

This mechanism allows the user to effect changes to the SCRIBE database formats which would ordinarily be imposed by SCRIBE.

There are several levels at which changes can be effected by appropriate use of the STYLE, @MODIFY or @DEFINE instructions, or by altering environment definitions as and when used by employing a modified form of the @BEGIN instruction.

The method of effecting these changes is now described along with the appropriate sets of keywords and their values.

There is a different set of keywords for use with the @DEFINE and @MODIFY instructions and the @STYLE instruction. Accordingly, the descriptions are split into two groups.

In the case of the @STYLE instruction, those keywords which may be used only at the beginning of an input file are so indicated.

6.1 The @STYLE Instruction

The simplest way of overriding the document formats specified in the SCRIBE database is to do so on a global basis, that is, specify some change which affects the format of the processed document as a whole, for example, specify double spacing where ordinarily single spacing would be the norm, or turn off justification where ordinarily it would be turned on, etc.

Such changes are effected with the @STYLE instruction which has the form:

@STYLE(..,**keyword value**,..)

where the **keyword value** pairs are respectively the overall document style attribute which is to be changed, and the value to which the attribute is to be changed, eg:

```
@STYLE[SPACING 2, INDENT 0, JUSTIFICATION OFF]
```

As the @STYLE instruction affects the overall document appearance, the @STYLE instruction must usually be used at the beginning of the manuscript file before any text to be processed.

There is a large set of **keyword value** pairs which may be specified with the @STYLE instruction, these are now described:

6.2 STYLE Keywords

BOTTOMMARGIN	specifies the vertical distance to be left between the last line of printing and the physical bottom of paper (beginning only), eg: `..,BOTTOMMARGIN 0.75 INCHES,..`
DOUBLESIDED	if **value** is set to YES, forces major headings and section titles onto odd pages if this option is available in the document type, eg: `..,DOUBLESIDED YES,..` If set, DOUBLESIDED activates the PAGEBREAK UNTILODD or PAGEBREAK UNTILEVEN attribute when encountered in a @DEFINE or @MODIFY instruction. If DOUBLESIDED is not set, PAGEBREAK BEFORE is assumed.
INDENT	specifies by how much to indent the first line of each paragraph in the text (beginning only). For example, to indent by 5 spaces give:

	`..,INDENT 5,..`
INDENTATION	same as INDENT.
JUSTIFICATION	specifies whether text is to be aligned along the right margin (YES or ON) or left unaligned (NO or OFF), eg: `..,JUSTIFICATION OFF,..` The JUSTIFICATION keyword may be used at the beginning of the manuscript file only.
LEFTMARGIN	specifies the width of the left margin - the horizontal distance between the physical edge of paper and the global left margin (beginning only), eg: `..,LEFTMARGIN 1 INCH,..`
LINEWIDTH	specifies the width of a print line - the horizontal distance between the global left margin and the end of the line (beginning only), eg: `..,LINEWIDTH 60,..`
NOTES	specifies the style in which notes are to be incorporated in the processed document. The value of NOTES should be FOOTNOTE for footnotes, ENDNOTE for end-notes, and INLINE for notes placed in line with the running text (suitable for unpaged devices such as FILE. The default value is FOOTNOTE.
PAGENUMBER	allows the page number style to be specified, eg: `..,PAGENUMBER "-@1-",..`

The string "@1" in this context is known as a **template**. Briefly, templates may be used to control the style of numbering of any sectioning instruction, numbered environment or counter etc. The templates available are:

@1	Arabic cardinal numerals
@A	upper case letters
@a	lower case letters
@I	upper case Roman numerals (1-5000)
@i	lower case Roman numerals (1-5000)
@O	capitalised English cardinals (One, Two, Three, etc.)
@o	uncapitalised English cardinals (one, two, three, etc.)
@F	capitalised English ordinals (First, Second, Third, etc.)
@f	uncapitalised English ordinals (first, second, third, etc.)
@'	Arabic ordinals (1st, 2nd, 3rd, etc.)
@*	sequence of asterisks (1-10 only)

Do not confuse the @I numbering template with the @I facecode instruction - although the form is the same, one is a keyword, one is an instruction and they are used in different situations.

For a fuller discussion of templates and the way in which they may be used to modify the way in which the sectioning instructions, enumerated environments and counters are numbered, please consult the SCRIBE User Manual [8].

PAPERLENGTH	a vertical distance specifying the paper length. Only meaningful for those devices with different paper sizes (beginning only), eg: `..,PAPERLENGTH 11.5 INCHES,..`
PAPERWIDTH	a horizontal distance specifying the paper width. Only

	meaningful for those devices with different paper widths (beginning only), eg: `..,PAPERWIDTH 9.25 INCHES,..`
RIGHTMARGIN	specifies the width of the right margin - the horizontal distance between the global right margin and the physical edge of paper, (beginning only), eg: `..,RIGHTMARGIN 4 INCHES,..`
SINGLESIDED	turns off DOUBLESIDED if set. SINGLESIDED takes no **value** keyword, eg: `..,SINGLESIDED,..`
SPACING	specifies the vertical distance from the base of one line to the base of the next line (beginning only). For example, to get double spacing rather than the default single spacing, give: `..,SPACING 2,..`
SPREAD	a vertical distance, added to SPACING to increase the inter-paragraph spacing (beginning only), eg: `..,SPREAD 0,..`
STRINGMAX	the maximum number of characters allowed in a delimited string in the manuscript file, eg: `..,STRINGMAX 100,..` The default value of STRINGMAX is 1024.
TOPMARGIN	specifies the amount of white space between the physical top of paper and the first line of printing on the page, eg: `..,TOPMARGIN 1 INCH,..`

Any Unilogic STYLE keywords not described here may be assumed not to have been implemented in the local version of SCRIBE.

6.3 The @MODIFY Instruction

The next level at which changes to a document format may be effected is by changing the appearance of a named environment. This is achieved by using the @MODIFY instruction which has the form:

@MODIFY[**environment,..keyword value,..**]

where **environment** is the name of the environment to be modified, and the **keyword value** pairs are the attributes of the environment which are to be modified and their modified values eg:

```
@MODIFY(TEXT, JUSTIFICATION OFF, LEFTMARGIN 2 INCH, RIGHTMARGIN 2 INCH)
```

The effect of the @MODIFY instruction is to alter the appearance of the named environment as specified in the list of **keyword value** pairs. This effect will apply during the whole processing of that particular manuscript file (unless further overridden by another modifying instruction).

As with the @STYLE instruction, the environment keywords specify a particular attribute or property of the environment, and the associated value word defines the value of that property. In some cases, there is a default value for a keyword if it is not mentioned, and another, different value if it is mentioned in a @MODIFY (or @BEGIN or @DEFINE) instruction.

6.4 Modification with the @BEGIN Instruction

Additionally, environments may be modified by the inclusion of keyword-value pairs in the @BEGIN instruction when the environment is used in the form:

```
@BEGIN(environment)
   .
   .
@END(environment)
```

This is done by modifying the @BEGIN instruction by inserting **keyword=value** pairs after the environment name, thus:

```
@BEGIN(environment,keyword value,..)
```

Such a modification will last for the duration of that particular excursion into the environment only, eg:

```
@BEGIN<FLUSHLEFT, SPACING 2, LEFTMARGIN 0>
   .
   .
@END[FLUSHLEFT]
```

6.5 The @DEFINE Instruction

The final way of effecting changes to the finished document is to define and use totally new environments. This type of change is achieved by using the @DEFINE instruction which may take one of two forms:

```
@DEFINE{newenvironment,..,keyword value,..}
@DEFINE"newenvironment=oldenvironment,..keyword value,.."
```

where **keyword** and **value** have their usual meanings, **newenvironment** is the name of the environment being defined, and **oldenvironment** is the name of an existing environment.

The effect of the first form is to define a totally new environment according to the attributes specified in the **keyword value** list, eg:

```
@DEFINE'RAGGEDTEXT, FILL, JUSTIFICATION OFF, SPACING 1'
```

The effect of the second form is to equate **newenvironment** to **oldenvironment** and then to modify **newenvironment** as specified in the **keyword value** list, eg:

```
@DEFINE[RAGGEDTEXT=TEXT, JUSTIFICATION OFF]
```

6.6 Environment Keywords

The set of keywords and values for use with the @MODIFY, modified @BEGIN and @DEFINE instructions is different from the set for use with the @STYLE instruction (though there is some duplication).

These keywords are now described.

ABOVE — specifies the amount of blank space left above an environment. Units should be a vertical distance, eg:

```
..,ABOVE 3,..
..,ABOVE 0.5 INCHES,..
```

If a unit 'name' (eg: INCHES) is omitted, LINES is assumed.

AFTERENTRY — takes a quoted instruction or string as its value. This instruction is evaluated and inserted into the input on entry to the environment, eg:

```
..,AFTERENTRY "@TABCLEAR",..
```

`BELOW`

specifies the amount of blankspace left below an environment. Units are as for ABOVE, eg:

```
..,BELOW 2 CM,..
..,BELOW 4 LINES,..
```

`BLANKLINES`

specifies what is to be done with blanklines found in an environment. **value** may take the value IGNORED, BREAK or KEPT, eg:

```
..,BLANKLINES KEPT,..
```

`BREAK`

controls the output of line breaks at the start and finish of environments. **value** may be one of BEFORE, AFTER, AROUND or OFF, eg:

```
..,BREAK AROUND,..
```

If the BREAK keyword is not present, BREAK OFF is assumed. If the BREAK keyword is present, the default value is AROUND.

`CAPITALIZED`

controls whether output text is in upper case. **value** should be one of ON, YES, OFF, NO, eg:

```
..,CAPITALIZED NO,..
..,CAPITALIZED ON,..
```

`CENTRED`

specifies that text be placed centrally between the prevailing margins. CENTRED does not take a value word, eg:

```
..,CENTRED,..
```

CENTRED is one of a group of mutually exclusive environment attributes - CENTRED, FILL, FLUSHLEFT and FLUSHRIGHT.

`CONTINUE`

if BREAK AROUND or BREAK AFTER is set, determines whether the line following an excursion into another environment resumes the previous paragraph or starts a new one. **value** may be any of ALLOWED, FORCE or OFF, eg:

```
..,CONTINUE ALLOWED,..
```

`COUNTER`

specifies which counter applies to the environment, and which counter is found by @TAG. **value** should be one of the counter names, eg:

```
..,COUNTER CHAPTER,..
```

`FACECODE`

controls the facecode (typeface) in which the environment is printed, eg:

```
..,FACECODE I,..
```

The FACECODE keyword should only be used when the required facecode (typeface) is not already in use for one of the facecode environments (in which case the USE keyword should be used to select the typeface - see below). Do not confuse the facecodes (typefaces) with the facecode environments (see section 5.5 - any correspondence between facecode environment 'name' and facecode 'name' is coincidental, and should not be regarded as a reliable basis for a 'rule' for selecting typefaces. For example on the X2700 laser printer, there is a facecode environment U, but no corresponding facecode U. For a list of the facecodes supported by the various devices accessible to SCRIBE, please see section 10.

FILL
forces line filling. FILL takes no value word. CENTRED, FILL, FLUSHLEFT and FLUSHRIGHT are mutually exclusive attributes, eg:

```
..,FILL,..
```

FLUSHLEFT
forces each line of text to the current left margin. FLUSHLEFT takes no value word, and CENTRED, FILL, FLUSHLEFT and FLUSHRIGHT are mutually exclusive attributes, eg:

```
..,FLUSHLEFT,..
```

FLUSHRIGHT
forces each line of text to the current right margin. FLUSHRIGHT takes no value word, and CENTRED, FILL, FLUSHLEFT and FLUSHRIGHT are mutually exclusive attributes, eg;

```
..,FLUSHRIGHT,..
```

FONT
selects a 'font' other than the default (usually called BODYFONT), on those devices which have other fonts defined, either in the database or by the user, eg:

```
..,FONT MYFONT,..
```

It is essential that the mechanism by which typeface names are assigned to facecode names with the @DEFINEFONT instruction, and by which facecode environments are defined with the @DEFINE instruction are fully understood before attempting to use fonts other than the default. This understanding will only be gained by consulting the relevant portions of the Unilogic manual [8].

FREE
turns 'off' the GROUP attribute (see below) if set. This may be necessary with some environments (eg: DISPLAY, EXAMPLE) where GROUP is usually turned 'on' in the database, and where this attribute is **not** required, eg:

```
..,FREE,..
```

GROUP
tests to see if there is enough space left on the current page to accommodate the text contained in this excursion into the environment. If not, the text is forced to the top of the next page, eg:

```
..,GROUP,..
```

The GROUP attribute is used in the GROUP environment (see section 5.7).

INDENT
controls 'start of paragraph' indentation with respect to the current left margin. **value** should be a horizontal distance, eg:

```
..,INDENT 0.25 INCH,..
..,INDENT 6,..
```

If no unit is specified, CHARS is assumed.

INDENTATION
same as INDENT.

INITIALIZE
same as AFTERENTRY.

JUSTIFICATION
controls right justification of the output, and may only be used in a filled environment. **value** should be one of ON, OFF, YES or NO, eg:

```
..,JUSTIFICATION YES,..
..,JUSTIFICATION OFF,..
```

LEADINGSPACES

controls what is done with leading spaces in manuscript file lines. **value** should be one of IGNORED, COMPACT or KEPT, eg:

```
..,LEADINGSPACES KEPT,..
```

LEFTMARGIN

sets the width of the left margin. **value** should be a horizontal distance, eg:

```
..,LEFTMARGIN 5,..
..,LEFTMARGIN +0.5 INCHES,..
```

Signed distances are effected with respect to the prevailing left margin; unsigned distances are effected with respect to the left hand edge of the output file 'page' (and thus will produce a left margin of that size when the output file is printed). LEFTMARGIN, LINEWIDTH and RIGHTMARGIN are related keywords, and only two of them may be specified at any one time.

LINEWIDTH

specifies the distance between the left and right margins. A horizontal distance should be used, eg:

```
..,LINEWIDTH 65 EMS,..
..,LINEWIDTH 7 INCHES,..
```

LEFTMARGIN, LINEWIDTH and RIGHTMARGIN are related keywords, and only two of them may be specified at any one time.

NEED

enables a test for sufficient space remaining on the current page to be made. **value** should be a vertical distance, eg:

```
..,NEED 4 LINES,..
..,NEED 3.5 INCHES,..
```

If there is not enough space remaining on the current page, a page throw will be inserted in the output file. Note that the NEED keyword and the @NEED instruction (see section 3.2 are **not** the same.

NUMBERED

specifies a template for numbering paragraphs in the environment, eg:

```
..,NUMBERED "-@1-",..
```

Please consult the SCRIBE User Manual [8] for further details.

NUMBERFROM

gives the initial value for the numbering of paragraphs in an environment, eg:

```
..,NUMBERFROM 4,..
```

PAGEBREAK

controls whether the start or end of an environment forces a page break. **value** may be any of the set; OFF, BEFORE, AFTER, AROUND, UNTILODD or UNTILEVEN, eg:

```
..,PAGEBREAK UNTILODD,..
```

The setting of the DOUBLESIDED keyword with a @STYLE instruction may have a bearing on the effect of a PAGEBREAK keyword. Please see sections 6.1 and 6.2 for details.

REFERENCED

is like NUMBERED but provides a template for cross-references to the paragraph number, eg:

```
..,REFERENCED "see section @1",..
```

Please consult the SCRIBE User Manual [8] for further details.

RIGHTMARGIN

sets the width of the right margin. **value** should be a horizontal distance, eg:

```
..,RIGHTMARGIN -5,..
```

	`..,RIGHTMARGIN 0.5 INCHES,..` Signed distances are effected with respect to the prevailing right margin; unsigned distances are effected with respect to the right hand edge of the output file 'page' (and thus will produce a margin of that size when the output file is printed). LEFTMARGIN, LINEWIDTH and RIGHTMARGIN are related keywords, and only two of them may be specified at any one time.

SCRIPT	specifies whether text is to be placed above or below the global baseline. **value** may be either +1 or −1, eg:
	 `..,SCRIPT +1,..`
	 `..,SCRIPT -1,..`

SPACES	specifies how multiple spaces in the manuscript file are treated. **value** should be one of KEPT, COMPACT, IGNORED or NORMALIZED, eg: `..,SPACES COMPACT,..`
SPACING	sets the line spacing. **value** should be a vertical distance, eg: `..,SPACING 0.25 INCHES,..`
SPREAD	specifies the additional vertical space to be added to SPACING when setting up the inter-paragraph spacing. **value** should be a vertical distance, eg: `..,SPREAD 1,..`
UNDERLINE	specifies which characters are to be underlined in the document file. **value** should be one of OFF, ALPHANUMERICS, NONBLANK or ALL, eg: `..,UNDERLINE ALL,..`
UNNUMBERED	states that an environment is to have unnumbered paragraphs. UNNUMBERED does not take a value word, eg: `..,UNNUMBERED,..`
USE	allows one environment definition to be used within another, eg: `..,USE B,..` This is the simplest way of changing the typeface used for a given environment, but please see the SCRIBE User Manual [8] for further details.

Any Unilogic environment keywords not described here may be assumed not to have been implemented in the local version of SCRIBE.

6.7 Page Layout

Figure 6.7 shows the notional page layout used by SCRIBE in formatting the standard environments described in section 5 in terms of those keywords which affect the overall page layout. The use or effect of many keywords which produce only a slightly 'visible' effect on overall page layout is not illustrated.

The capitalized words in this figure are SCRIBE keywords which are described in the appropriate sections above (sections 6.2 and 6.6).

The instruction used to set the value of any given **keyword** is indicated in brackets after the keyword, according to the following scheme: D − @DEFINE instruction; M −

@MODIFY instruction; S - @STYLE instruction. These three instructions are described in detail in sections 6.1, 6.3 and 6.5 above.

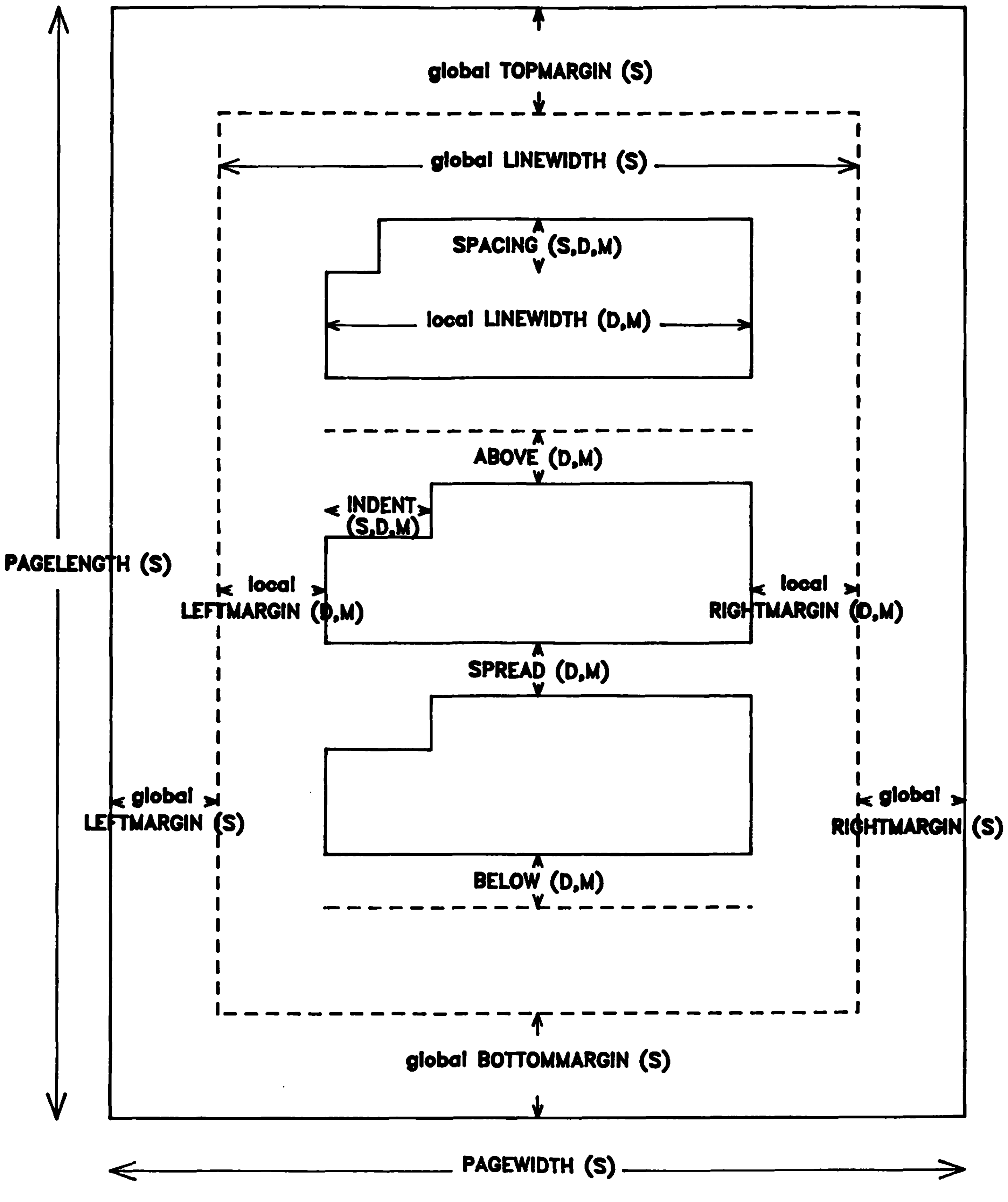

Figure 6.7: Nominal Page layout used by SCRIBE

7 Predefined Strings

The `@VALUE` instruction retrieves the contents of strings and outputs them to the document file. These strings can either have been defined by the user, using the `@STRING` instruction or can be one of a number of strings predefined and known to SCRIBE. The form of the `@VALUE` instruction is:

`@VALUE(`**stringname**`)`

and the following predefined stringnames yield the specified results:

7.1 Site Information

SCRIBEVERSION	The current SCRIBE version number, eg: `3.6`
SITE	The name of the institution running SCRIBE, eg: `E.R.C.C.`

7.2 Calendar Information

DATE	The current calendar date; for example `21/11/86`
DAY	The day of the month, eg: `21st`
MONTH	The name of the current month of the year, eg: `November`
TIME	The time when the current SCRIBE run began, eg: `16.15.13`
WEEKDAY	The name of the current day of the week, eg: `Friday`
YEAR	The current year, eg: `1986`

7.3 Information on Current SCRIBE Run

DEVICENAME	The name of the output device requested for this run, eg: `Xerox 2700`
MANUSCRIPT	The name of the manuscript file currently being processed, eg: `SCRIBE_MANUAL`
PAGE	The current page number in this document, eg: `27`
SECTIONTITLE	The section title specified in the last sectioning instruction, eg: `Information on Current SCRIBE Run` In an unsectioned document a null value is returned.
SECTIONNUMBER	The section number of the last sectioning instruction given, eg:

	`7.3`
	In an unsectioned document a null value is returned.
TIMESTAMP	The date and time when the current SCRIBE run began, eg:
	`21/11/86 16.15.13`
USERNAME	The user account (or job) code is given, eg:
	`ERCY02`

Any Unilogic predefined strings not described here may be assumed not to have been implemented in the local version of SCRIBE.

8 Document Types

8.1 Introduction

The standard environments described above are always available as is the automatic counting and numbering of pages. Depending on the **documenttype** specified in the @MAKE instruction:

@MAKE(**documenttype**)

there may be additional document-specific environments, counters, cross-referencing and indexing facilities available.

For example, document type LETTER has an environment defined for each of the parts of a personal letter - the address, the greeting, the body, etc.; (there is no requirement that these environments be used in a sensible order, or indeed at all).

Other document types provide:

- automatic counting and numbering of sections, and of subsections within sections, etc., to a variety of levels according to document type
- cross referencing within the text to preceding or following sections
- automatic contents page generation in sectioned documents
- automatic counting and numbering of theorems, equations etc., within sections
- automatic indexing in several styles

The following are the document types available in SCRIBE:

8.2 Unsectioned Document Types

LETTER produces a personal letter. The following additional environments are defined:

- RETURNADDRESS
- ADDRESS
- GREETING
- BODY
- NOTATIONS (closings or initials)
- POSTSCRIPT

The RETURNADDRESS environment is entered automatically at the start of the manuscript.

TEXT is the default document type (ie, the one SCRIBE uses if there is no

@MAKE instruction in the manuscript file). It has no facilities beyond the standard ones. At the start of the manuscript it enters the TEXT environment mentioned earlier, which has filled, justified paragraphs. TEXT is the only document type which is also an environment and it is both the default environment and the default document type. That is, if no instructions indicate otherwise, SCRIBE will behave as if it had encountered the three instructions:

```
@MAKE(TEXT)
@BEGIN(TEXT)
    .
    .
@END(TEXT)
```

whenever it processes a manuscript file.

8.3 Sectioned Document Types

ARTICLE is the simplest of the sectioned document types. The following sections are defined (please see section 3.5 for full details):

- SECTION
- SUBSECTION within SECTION
- PARAGRAPH within SUBSECTION
- APPENDIX
- APPENDIXSECTION within APPENDIX
- PREFACESECTION (unnumbered)
- UNNUMBERED section (unnumbered)

All sections are automatically numbered except where indicated. ARTICLE has an automatically generated table of contents but no index.

REPORT is a sectioned document type similar to ARTICLE but which has numbered CHAPTERs in addition to the sections available in ARTICLE.

MANUAL is a sectioned document type which recognises the same sectioning instructions as REPORT but also has indexing facilities (see section 3.8).

Other document types familiar to the experienced SCRIBE user, THESIS, BROCHURE, GUIDE, REFERENCECARD and SLIDES are not yet available with SCRIBE. It is intended that there will be a local EMAS document type called VIEW which will produce a document file suitable for "Viewing" on-line.

8.4 Additional Text Environments

There are three additional 'mathematical' environments available with the sectioned document types. These include THEOREM, LEMMA and EQUATION. These are now described:

THEOREM allows for the statement of a theorem. The theorems will be numbered sequentially (1, 2, 3, etc.) and the numbering sequence will be restarted at 1. each time a new section is started, eg:

```
@BEGIN(THEOREM)
The sum of the square on the hypotenuse is equal to the sum
of the squares on the other two sides.
@END(THEOREM)
```

THEOREM is a filled environment.

LEMMA allows for the statement of a lemma. Lemmas are numbered concurrently with any theorems which may be used, and will therefore also have their numbers reset to 1. each time a new section is started, eg:

```
@BEGIN<LEMMA>
Let the nodes of a non-empty binary tree be
 u@-[1],...,u@-[n]
in preorder, and let f(u)=l(u)+r(u)-1. Then:
   .
   .
   .
@END<LEMMA>
```

LEMMA is a filled environment.

EQUATION allows for the statement of an equation. Equations are not numbered unless they are TAGged (see section 3.9). When so numbered, they are numbered sequentially, and independently of the section in which they are encountered, eg:

```
@BEGIN(EQUATION)
f(u@-[1])+f(u@-[2])+...+f(u@-[n])=-1, and
f(u@-[1])+...+f(u@-[k])>=0 for 1=<k<n@TAG[EQNP]
@END(EQUATION)
```

EQUATION is an unfilled environment.

Other 'mathematical' environments include PROOF, DEFINITION and COROLLARY.

9 Device Types

The device type for which the user wishes a manuscript file to be formatted is specified by the @DEVICE(**devicename**) instruction. If none is specified SCRIBE prepares the output for the normal computer line printer. The characteristics of the various devices recognised by SCRIBE are as follows:

9.1 Computer Line Printer

SCRIBE device type	LPT
Page length	66 lines
Page width	132 characters
Devices available on EdNet	.LP25, .LP15, all 'standard' line printers

Only one typeface is available with this device. Boldfacing is achieved by overprinting, and sub- and super-scripting are achieved by using a full line for the sub- or super-script.

9.2 Unpaged Terminal File

SCRIBE device type	FILE
Page length	Unpaged
Page width	80 characters
Devices available on EdNet	all 'standard' terminals

Only one typeface is available with this device. Underlining, overprinting, sub- and super-scripting are not possible with this device.

9.3 Philips GP300 Dot Matrix Printer

SCRIBE device type	GP300
Page length	11.6 inches
Page width	8.25 inches
Devices available on EdNet	.DP15, .DP25

The typefaces (facecodes) available with this printer are described briefly in section 10.2 below. For a full description, please consult ERCC User Note 50 [4].

There is an irreducible hardware topmargin of 0.5 inches, and hardware left margin of 0.75 inches associated with this device. These values must be taken into account whenever any modification is being considered.

The line spacing may only be set in integral multiples of 0.25 lines. Line spacings not conforming to this requirement will be rounded down to the next line spacing which does satisfy this condition; thus `SPACING 1.26` will be rounded to `SPACING 1.25`, and `SPACING 1.24` to `SPACING 1.00`.

Warning: A real money charge may be made on a per sheet basis for using this device. For details of current charges, please consult the ERCC Advisory service.

A 'dummy' drafting device type known to SCRIBE as `DRAFTGP300` may also be used. This device type may be used to prepare drafts of documents which are ultimately to be listed on a Philips GP300 printer. These drafts will have the same page structure as the final document, but may be listed at a standard line printer (**not** true of documents prepared for device `GP300`), thus giving a cheap, quick drafting facility.

9.4 Xerox X2700 Laser Printer

SCRIBE device type	X2700	
Page length	11.6 inches	11.6 inches
Page width	8.25 inches	16.5 inches
Devices available on EdNet	.DP23	

This device allows the choice of two paper sizes - effectively these are A4 'portrait' and A3 'landscape'. Both of these paper sizes employ the 'portrait' mode typefaces resident in the printer - **not** the 'landscape' mode typefaces - these are not currently available within SCRIBE.

A4 'landscape' mode can be simulated by using A3 paper size and making the necessary adjustments in the `PAPERWIDTH` and `LINEWIDTH` settings in the manuscript file, and once the document has been listed to the appropriate forms queue (`40` for A3 paper) trimming the output down to A4 'landscape' format manually. For full details of the `LIST` command, please consult the relevant EMAS User's Guide [2], [3].

The typefaces (facecodes) available with this printer are described briefly in section 10.3 below.

Line spacing on this device is a function of the typeface being used - thus changing to a different typeface should not require the line spacing to be reset; this will happen automatically.

Warning: A real money charge may be made on a per sheet basis for using this device. For details of current charges, please consult the ERCC Advisory service.

A 'dummy' drafting device type known to SCRIBE as DRAFTX2700 may also be used. This device type may be used to prepare drafts of documents which are ultimately to be listed on a Xerox X2700 or compatible laser printer. These drafts will have the same page structure as the final document, but may be listed at a standard line printer (**not** true of documents prepared for device X2700), thus giving a cheap, quick drafting facility.

10 Device Typefaces

10.1 Line Printer Typefaces

The line printer (device LPT) has only one fixed width typeface. Use of the FACECODE environment attribute will therefore generally result in a warning message being output by SCRIBE (see section 11.2).

10.2 Philips GP300 Typefaces

The Philips GP300 dot-matrix printer has a total of 30 typefaces, of which 25 are available via SCRIBE at the moment. The different typefaces are assigned to a facecode 'name' consisting of a letter of the alphabet on an arbitrary basis, except that those typefaces which are used in the definition of SCRIBE's standard facecode environments (see section 5.5) usually have the same facecode and facecode environment name, that is, on the GP300 for example, the typeface used as standard for running text is Letter Gothic; this typeface is assigned to the (facecode) name R in the database, and this (facecode) name is subsequently used in the definition of the (facecode) environment R.

A	**Letter Gothic Bold 12pitch.**
B	**Letter Gothic Bold proportionally spaced.**
C	Not used.
D	**Letter Gothic Bold 10 pitch.**
E	Courier 10 pitch.
F	Courier 12 pitch.
G	Γρεεκ προπορτιοναλλψ σπαχεδ (Greek proportionally spaced).
H	Courier proportionally spaced.
I	*Letter Gothic Italic proportionally spaced.*
J	**Micro 10 pitch.**
K	**Micro 15 pitch.**
L	***ORATOR 12 PITCH.***
M	ΝαυθεναυιγαμδΤγιεξυιζιγ ποοποουιοξαμμ™ τπαγεδ (Mathematical/Scientific proportionally spaced).
N	***ORATOR 10 PITCH.***
O	***ORATOR PROPORTIONALLY SPACED.***
P	*Letter Gothic Italic 12 pitch.*
Q	Data 10 pitch.
R	Letter Gothic proportionally spaced.
S	**Micro 12 pitch.**
T	Letter Gothic 10 pitch.
U	Letter Gothic 12 pitch.
V	*Letter Gothic Italic 10 pitch.*
W	Γρεεκ 12 πιτχ• (Greek 12 pitch).
X	Γρεεκ 10 πιτχ• (Greek 10 pitch).
Y	ΝαυθεναυιγαμδΤγιεξυιζιγ |r πιυγθ (Mathematical/Scientific 12 pitch).
Z	ΝαυθεναυιγαμδΤγιεξυιζιγ |— πιυγθ (Mathematical/Scientific 10 pitch).

The full character set for each typeface is listed in ERCC User Note 50 [4].

10.3 Xerox X2700 Typefaces

The Xerox X2700 laser printer has a total of 43 typefaces, all of which are available within SCRIBE. In the 'standard' font (known as BODYFONT) each typeface is assigned to a facecode 'name' consisting of a letter of the alphabet on an arbitrary basis, except that those typefaces which are used in the definition of SCRIBE's standard facecode environments (see section 5.5) usually have the same facecode and facecode environment name, that is, on the X2700 for example, the typeface used as default for running text is Kosmos 10; this typeface is assigned to the (facecode) name R in the 'standard' font, and this (facecode) name is subsequently used in the definition of the (facecode) environment R. In addition, where the character set associated with a given typeface extends over more than 96 characters, the 'extended' range of characters is assigned the same facecode name with the addition of the digit 1 - thus, the extended range of characters for the Kosmos 10 typeface is assigned the name R1 (in the 'standard' font).

A selection of the available typefaces (with their 'standard' facecode names and the facecode and text environments which employ them) are:

SCRIBE typeface name	'Standard' facecode name	Sample (and environments where used)
KOSMOS10B	B	**Kosmos 10 point bold, proportionally spaced.** (B, HEADING, SUBHEADING)
COMPMODROMAN11	D	Computer Modern Roman 11 point, proportionally spaced.
COMPMODITALIC11	E	*Computer Modern Italic 11 point, proportionally spaced.*
TITAN12	F	Titan 12 pitch, fixed width. (T, VERBATIM, EXAMPLE, PROGRAMEXAMPLE)
GREEK10	G	Γρεεκ 10 ποιντ, (Greek 10 point). (G)
KOSMOS10I	I	*Kosmos Italic 12 point, proportionally spaced.* (I)
KOSMOS12B	K	**Kosmos 12 point bold, proportionally spaced.**
KOSMOS14	L	Kosmos 14 point, proportionally spaced.
MATHS10	M	∂ατησ 10 ρ∔ιντ, (Maths 10 point). (M)
KOSMOS10	R	Kosmos 10 point, proportionally spaced. (R, TEXT, ITEMIZE, ENUMERATE, VERSE, FORMAT, FLUSHLEFT etc.)
KOSMOS8	S	Kosmos 8 point, proportionally spaced. (+, −, C)

It must be stressed that these facecode assignments apply only to the standard font (or selection of typefaces), which is known as BODYFONT. The informed user may define their own fonts using the @DEFINEFONT instruction which may then be selected with the FONT keyword in any of the valid ways. This will in general result in a different typeface being associated with each facecode name, and consequently with each (facecode or text) environment.

A User Note is in preparation illustrating the full range of typefaces in this font, and which also illustrates the range of typefaces employed in the definition of several alternative fonts, which allow the selection of a different range of typefaces, resulting in all environments being printed in typefaces other than those defined in BODYFONT. This selection is achieved with the FONT keyword (see section 6.6), and is an easier mechanism for changing typefaces than doing so on an individual basis using the FACECODE keyword (see section 6.6).

Users may define their own fonts using the @DEFINEFONT instruction (see section 3.7), but are urged to seek advice, or consult the Unilogic documentation [9] in order to achieve a proper understanding of the mechanisms involved before so doing.

11 Messages from SCRIBE

SCRIBE has sophisticated error handling routines and is capable of producing messages to cover a wide range of occurrences.

Error messages are displayed on the terminal at run time and are also stored in a file called **manuscript**£ERR where **manuscript** is the name of the manuscript file, or the first six characters of the manuscript file name if it is greater than six characters long. This file is stored on the user's file store and may be inspected at leisure with any suitable editing facility. If no errors are detected, no error file is produced.

The messages are split very broadly into three types; fatal errors which will result in the job being abandoned, non-fatal errors which will result in the run being completed but which may result in unpredictable results in the output file, and information messages.

These are now described:[3]

11.1 Fatal Error messages

These messages are produced when fatal errors are encountered. SCRIBE will abandon processing and no output will be produced.

```
disaster in endnote buffer.
a footnote or group exceeds the pagelength.
run abandoned.
```

SCRIBE has encountered a footnote or a group which is longer than one page. This is not allowed. The probable reason for this error occuring is that a footnote or group has not been closed correctly with a closing bracket, please check your manuscript file carefully.

[3]Messages from Unilogic SCRIBE are different from those described here.

```
disaster in generated text arising from line n of file manuscript :-
ran off text in directive processing, missing terminator?
run abandoned.
```

A missing closing delimiter has been detected in one of the SCRIBE instructions which generates text (eg: @STRING). Check for missing delimiters.

```
disaster in line n of file manuscript :-
no font set for the base environment.
run abandoned.
```

The database contains no specification for the default typeface for the selected font. This is a database problem - please consult the ERCC Advisory service.

```
disaster in line n of file manuscript :-
no string space for codewords.
run abandoned.
```

The total space available for @REF codewords has been exhausted. You should either remove or shorten some of your codewords, or contact the ERCC Advisory service to report the problem.

```
failed to connect file library file.
```

One of the four database files is missing from the SCRIBE database being used. The values of **file** which may appear are DEVICE, MAKE, TYPECASE or WIDTH. If you are using the system database, please consult the ERCC Advisory service. If you are using a private database, please consult the database maintainer.

```
disaster in line n of file manuscript :-
page frame overflow
run abandoned.
```

A too large page size has been selected in a @DEFINE, @STYLE or @MODIFY instruction.

```
disaster in line n of file manuscript :-
ran off text in directive processing. missing terminator?
run abandoned.
```

An instruction has been encountered which is incomplete in some unspecified way - often this will be a missing closing bracket.

```
disaster in line n of file manuscript :-
string too large.
run abandoned
```

A delimited (quoted) string contained in (say) a @STRING instruction contains too many characters. The default is 1024 characters, and this value may be increased by using the @STYLE instruction and the STRINGMAX keyword to increase this value.

```
disaster in line n of file manuscript :-
too many input files.
run abandoned.
```

Too many files have been inserted in the manuscript file with the @INCLUDE instruction. The limit is 8.

```
disaster in line n of file manuscript :-
too many nested environments
run abandoned
```

The nesting of environments has been carried to too deep a level. The cause is probably a missing close to an environment.

```
disaster in line n of MAKE file :-
table of contents overflow
run abandoned
```

The table of contents generated is too large to fit into the available space. Please report to the ERCC Advisory service.

```
failed to locate entry for device in library file.
```

You have attempted to specify an unknown device in a @DEVICE instruction.

```
failed to locate entry for document in library file.
```

You have attempted to use an unknown document type in a @MAKE instruction.

```
failed to locate typecase typecase in library file.
run abandoned.
```

Reference has been made to a **typecase** (font) for which there is no entry in the database file. Check for incorrectly spelled typecase (font) names in the manuscript file, if these are all correct, please consult the ERCC Advisory service.

```
NEWSMFILE fails - File already exists.
```

This message is produced when a previous SCRIBE run has been aborted in the middle of the run, and appears because the temporary work file created by SCRIBE (**manuscript**£MS) is still in existence in your process. If the current job attempts to produce a larger temporary file than the one already existing, the run will fail. It is recommended therefore that if this message is encountered, the run is aborted and the file **manuscript**£MS DESTROYed before recommencing the run.

If the current SCRIBE run attempts to produce a smaller temporary file than the one already existing, the message will be produced, but the run will continue to completion.

```
NEWSMFILE fails - invalid parameter 0
```

This message will be produced if an attempt is made to process an empty manuscript file. SCRIBE will attempt to process the manuscript file and will then display a series of system error messages. You should check that you are processing the correct manuscript file.

```
no string space for string names
run abandoned.
```

The total space available for @STRING names has been exhausted. You should either remove or shorten some of your string names, or contact the ERCC Advisory service to report the problem.

```
the output file document is (one of) the input file(s) :-
run abandoned
```

The document file specified will over-write one of the manuscript files as SCRIBE processes the job. This is not allowed.

11.2 Non-Fatal and Warning Messages

These messages are produced when non-fatal errors are encountered. SCRIBE will continue processing and will produce an output file, but the contents will be unpredictable due to the error.

`error in generated text arising from line` **n** `of file` **manuscript** `:-`
`unknown directive -` **directive**

SCRIBE has encountered an inconsistency when generating text internally. This is probably due to an invalid, incorrect or missing closing delimiter in a sectioning instruction.

`error in generated text arising from page heading/footing on page` **n** `:-`
`line too long.`

SCRIBE has encountered an inconsistency when generating text internally. This is due to a too long page heading or footing.

`error in index buffer.`
`unknown directive -` **directive**

An unknown instruction has been encountered when processing the index. Please check your indexing instructions for the correctness of any embedded instructions. If these are all correct, there may be a problem with the database entries controlling the index format, please report the problem to the ERCC Advisory service.

`error in line` **n** `of file` **manuscript** `:-`
`bad parameter for VALUE -` **parameter**

An unknown parameter (name) has been encountered as the argument of a `VALUE` instruction. This is probably because a closing delimiter has not been supplied. Check the appropriate line of the manuscript file carefully.

`error in line` **n** `of file` **manuscript** `:-`
`bad RH side at` **text** `in FORM command`

The text form of the right hand side of a database `@FORM` instruction is in error in some way - probably due to a missing opening or closing delimiter in the SCRIBE database. **text** may comprise a lengthy string listing all subordinate forms within the `@FORM` instruction. Please consult the ERCC Advisory service.

`error in line` **n** `of file` **manuscript** `:-`
`bad RH side at` **text** `in TEXTFORM command`

The text form of the right hand side of a database `@TEXTFORM` instruction is in error in some way - probably due to a missing opening or closing delimiter in the SCRIBE database. Please consult the ERCC Advisory service.

`error in line` **n** `of file` **manuscript** `:-`
`bad tab -`

An unacceptable tab setting has been encountered in a `@TABSET` instruction, probably due to a distance or distance unit being mis-typed.

`error in line` **n** `of file` **manuscript** `:-`
`counter value exceeds the range for template code '`**t**`'. code '1' used.`

The value of a counter has exceeded the maximum allowed for the particular numbering style being used. The allowed range for the different numbering templates is given on page 18.

```
error in line n of file manuscript
END environment misplaced.
```

This message arises either because:

- An @END[**environment**] has been found which does not have a matching @BEGIN[**environment**], or because:
- A 'nested' environment has been encountered which does not have a closing @END[**environment**] instruction, thus making it appear that the @END instruction of the nesting environment is misplaced.

```
error in line n of file manuscript :-
illegal codeword - codeword for command command
```

An illegally long value has been provided for a @LABEL or @TAG instruction - the maximum length allowed is 31 characters.

```
error in line n of file manuscript :-
invalid dynamic LHM setting.
```

An erroneous left hand margin has been set with the @$ instruction, probably due to a tab boundary being exceeded.

```
error in line n of file manuscript :-
line too long.
```

Line **n** of the file **manuscript** occurs in an unfilled environment and is too long to fit onto the defined line width. The processed text will however override the defined line width, and an attempt will be made to format the line for the output file with unpredictable results.

```
error in line n of file manuscript :-
local margins inside-out - set to zero
```

The (probably user-modified) local margins of the relevant environment overlap, ie: the right hand margin is to the left of the left hand margin. Check the margin settings of any modified or user defined environments.

```
error in line n of file manuscript :-
misplaced DEVICE command ignored.
```

The @DEVICE instruction has not been placed before the text to be processed, or an additional @DEVICE instruction has been encountered.

```
error in line n of file manuscript :-
misplaced MAKE command ignored.
```

The @MAKE instruction has not been placed before the text to be processed, or an additional @MAKE instruction has been encountered.

```
error in line n of file manuscript :-
missing separator at keyword in command command
```

A keyword and its attribute are incorrectly separated in a @DEFINE, @MODIFY or @STYLE instruction.

```
error in line n of file manuscript :-
no 'referenced' template for counter
```

An attempt has been made to use @TAG within either @ITEMISE or @DESCRIPTION. @TAG may only be used within @ENUMERATE.

```
error in line n of file manuscript :-
no value defined for LABEL label
```

The numeric section value required for the @LABEL, @PAGEREF and @REF instructions to interact correctly has not been assigned a value - probably because an unsectioned document type is being used.

```
error in line n of file manuscript :-
no value defined for TAG tag
```

An attempt has been made to use @TAG outwith an itemising environment. @TAG may only be used within ENUMERATE.

```
error in line n of file manuscript :-
opening delimiter missing after command command
```

A command which requires a delimited (bracketed) argument has been used without an opening delimiter. Check your manuscript file at the line indicated.

```
error in line n of file manuscript :-
opening delimiter missing after form textform
```

A textform has been used without a delimited argument after the textform name. Please consult the ERCC Advisory service.

```
error in line n of file manuscript :-
return marker has not been set.
```

A return to marker instruction (@/) has been encountered when no return marker (@!) has been set. This is almost certainly because a return to marker (@/) instruction has been typed where a tab instruction (@\) was intended.

```
error in line n of file manuscript :-
right hand environment of newenvironment=oldenvironment is not defined.
```

A @DEFINE instruction is attempting to use a non-existent old environment in an attempt to set up a new environment.

```
error in line n of file manuscript :-
single character command c has not yet been implemented.
```

The single character instruction indicated by **c** is available in Unilogic SCRIBE, but not (yet) in the local implementations.

```
error in line n of file manuscript :-
style keyword keyword encountered after text start, ignored.
```

A STYLE instruction has been encountered after the start of the text to be processed - in most cases, this is forbidden, and the style setting indicated by **keyword** will not be implemented.

```
error in line n of file manuscript :-
tab set beyond right margin ignored.
```

A tab setting has been encountered which occurs beyond the current right margin. The setting will be ignored and the right margin itself will be used as a tab stop.

```
error in line n of file manuscript :-
the name - environment in command command is already defined as an environment.
```

You have attempted to redefine an already existing environment in a @DEFINE instruction, or equate one existing environment to another existing environment in an

@EQUATE instruction. These operations are forbidden.

```
error in line n of file manuscript :-
the name - name in command command is already defined as a codeword.
```

An attempt has been made to use the same **codeword** to label two (or more) @TAG and/or @LABEL instructions. This is not allowed, labels in @TAG and @LABEL instructions must be unique.

```
error in line n of file manuscript :-
undefined argument - argument for form parameter parameter
```

An invalid value has been supplied as an argument to either a @FORM or @TEXTFORM instruction. This is unlikely to be a user generated problem, and except in the specific case where the message appears in the form:

```
error in line n of file manuscript :-
undefined argument - PAGE for form parameter PARM
```

when document type FILE is being used, the message should be reported to the ERCC Advisory service.

```
error in line n of file manuscript :-
undefined environment for command command.
```

An attempt has been made to use an undefined environment in either a @BEGIN or an @END instruction.

```
error in line n of file manuscript :-
undefined string name for VALUE - stringname
```

An undefined STRING name has been used in a @VALUE instruction.

```
error in line n of file manuscript :-
unknown counter template code 'c'.
```

an unknown counter template (c) has been used when specifying the counter to be used in a numbered environment. Probable cause is an incorrect counter specification in an @DEFINE, @STYLE or @MODIFY instruction.

```
error in line n of DEVICE file :-
unknown DEFINEFONT keyword - key
```

An unknown facecode name has been used to assign a typeface name in a font definition in the database. This is a database problem, please consult the ERCC Advisory service.

```
error in line n of file manuscript :-
unknown DEFINEFONT keyword - key
```

An unknown facecode name has been used to assign a typeface name in a user-defined font in the manuscript file. Allowable facecode names are A,..,Z,A1,..Z1, check the names used in any user-defined font carefully.

```
error in line n of file manuscript :-
unknown directive - directive
```

An attempt has been made to use an instruction which is not recognised by SCRIBE. This error can arise in a number of ways:

- a genuinely erroneous instruction has been used

- a SCRIBE instruction which is inappropriate to the document type has been used, eg: counters and indexing instructions in document type TEXT

```
error in line n of file manuscript :-
unknown command keyword - keyword
```

An incorrect keyword has been used in a @STYLE, @DEFINE, @MODIFY, @BEGIN, @PAGEHEADING or @PAGEFOOTING instruction.

```
error in line n of file manuscript :-
unnecessary, missing or bad value for command keyword - keywordinerror
```

An attempt has been made to use an incorrect keyword or value in a @DEFINE, @MODIFY or @STYLE instruction.

```
error in page heading/footing on page n :-
```

The page heading or footing for page number **n** is too long. Check that the combined width of the three page heading/footing fields does not exceed the available line width.

```
error in table of contents buffer.
unnecessary, missing or bad value for PAGEHEADING/PAGEFOOTING keyword keyword
```

The table of contents page heading or footing is in error. This is likely to be a database problem, please consult the ERCC Advisory service.

```
error in table of contents buffer.
END string misplaced.
```

An error has arisen while generating section titles for the contents table. This is almost certainly because the section title concerned contains two similar closing delimiters. One of the pairs of delimiters should be changed to another type.

```
error in table of contents buffer.
no 'referenced' template for counter counter
```

The database entry for the named counter (usually a sectioning instruction) does not contain the necessary information to enable SCRIBE to construct the table of contents. Please contact the ERCC Advisory service.

```
error in table of contents buffer.
unknown directive - directive
```

SCRIBE has encountered an inconsistency in a section title when generating the table of contents (in valid document types). This is probably due to an invalid, incorrect or missing closing delimiter in a sectioning instruction.

```
error. the following codewords were referenced but not defined:-
  Codeword                Type     Value used    Line numbers(file)
```

Use has been made of a codeword in a @REF or @PAGEREF instruction which has not been defined in a @LABEL or @TAG instruction.

```
facecode x not defined in font y
base environment font and facecode used.
```

The specified facecode is not defined in the database for the selected device. This may be due to the fact that the device concerned is incapable of providing the facecode requested (eg: bold italic on a line printer). In any case, SCRIBE will 'do the best it can' with the text and will produce some form of output (eg: underlining instead of bold italic).

```
the following forward references were undefined:-
```

`line` **n** **label**

Reference has been made (in a @REF or @PAGEREF instruction) to a label which has not been defined with a @LABEL or @TAG instruction.

```
warning. a further run is required to obtain correct values for the following
forward references:-
  Codeword                     Type      Value used     Line numbers (file)
```

Either, @REF and @TAG codewords have been incorrectly counted due to a change in the file, or, on the first run with the manuscript file concerned, SCRIBE has not been able to correctly assign section values to all codewords - usually because of forward references to sections not yet processed. The solution is to process the manuscript file again.

```
warning. the following codewords were defined but not referenced:-
```

list
of
codewords

The listed codewords were defined in either a @LABEL or @TAG instruction, but never referenced in a @REF or @PAGEREF instruction.

11.3 Information Messages

These messages are for information only and some of them at least will be produced during any SCRIBE run.

`counter` **counter=n**

In sectioned documents, 'announces' which section is being processed. This is useful for monitoring the progress of a job. Counters so announced are, CHAPTER, SECTION, APPENDIX and APPENDIXSECTION.

`document produced in` **document**

SCRIBE has completed processing and stored the output in file **document**.

`phase` **n** `processing MS`

The manuscript file is being processed - either the initial instructions are being digested (**n**=1), or the main text is being processed (**n**=3).

`phase` **n** `processing library entry for` **device**

The database entry for the appropriate output device is being read.

`phase` **n** `processing library entry for` **documenttype**

The database entry for the appropriate document type is being read.

`phase` **n** `processing index.`

The index is being prepared in document types which allow this.

`phase` **n** `processing table of contents.`

the contents page is being processed in appropriate document types.

12 Additional Facilities

12.1 Extensions to the Basic SCRIBE Command

The basic SCRIBE command may be extended in several ways:

1. The manuscript file may be a member of a partitioned file, thus, the command:

   Command:SCRIBE **pdfile_member,document**

   where **pdfile_member** is the partitioned file member containing the input to SCRIBE is valid.
2. Manuscript files may be concatenated in a SCRIBE command to give one output file thus, the command:

   Command:SCRIBE **manuscript1+manuscript2+...+manuscriptn,document**

   is valid, where the **manuscripts** are the manuscript files to be processed with SCRIBE.
3. The output device may be specified at run time if a SCRIBE command of the form:

   Command:SCRIBE **manuscript,document,devicetype**

   is given, where **devicetype** is one of the device types known to SCRIBE (see section 9). If a device type has been specified with a @MAKE instruction in the manuscript file, this device is overridden by that specified at run time.
4. Personal databases may be referenced by a SCRIBE manuscript file if a modified form of the SCRIBE command is used:

   Command:SCRIBE **manuscript,document,,database**

   where **manuscript** and **document** have any of the meanings assigned to them in the SCRIBE primer [5] or at items 1 and 2 above, and **database** is the user's (or departmental, or whatever) personal database.

   A SCRIBE database is a pdfile with four members named DEVICE, MAKE, TYPECASE and WIDTH. It is envisaged that eventually, departments may wish to set up their own databases containing formatting information satisfying departmental requirements for certain types of documents. Setting up such databases is **not** a trivial task, and anyone intending to do this is recommended to contact in the first instance the ERCC Advisory staff.

12.2 Access to the ERCC VAX/VMS

Under certain circumstances, demanding users requiring to process very large multiple input file SCRIBE jobs may need direct access to the implementation of Unilogic SCRIBE running on the ERCC VAX/VMS machine. Details of how to run SCRIBE 'live' on this machine is outwith the scope of this document.

For details of how to gain direct access to this machine, please contact in the first instance the ERCC Advisory service.

Remember that if you do use Unilogic SCRIBE, your **manuscript** must contain Unilogic SCRIBE instructions only. There are one or two Edinburgh SCRIBE instructions which either do not exist in Unilogic SCRIBE (eg: @NEED), have a slightly different format (eg:

@INDEXSECONDARY) or are spelt differently (eg: Edinburgh SCRIBE - CENTRE; Unilogic SCRIBE - CENTER). In all cases of dubiety, the Unilogic SCRIBE User Manual [8] should be regarded as giving the correct information, though **most** of what is available in Edinburgh SCRIBE can in fact be processed with Unilogic SCRIBE without any change. Reference copies of the Unilogic manual are available for inspection in the ERCC Advisory offices at 59 George Square and the James Clerk Maxwell Building, and copies may be purchased at cost price from the same locations.

12.3 Keeping up-to-date

It is intended that both this document and the SCRIBE primer [5] be maintained in as up-to-date a state as is practicable. However, for the most recent additions/corrections/bug reports/bug fixes/etc. please see the electronic bulletin board SCRIBE. Bulletin boards are an EMAS MAIL facility which are fully described in the MAIL manual [6].

In addition, a **draft** copy of the next release of this manual is always available on-line. This manual is formatted for the line printer, and includes a description of new SCRIBE facilities which may not be described in the current hard-copy edition of the manual. The on-line manual may be listed with the command:

```
Command:LIST ERCY02.SCRIBEX_UN67,device
```

where **device** is a suitable output device (line printer).

13 References and Further Reading

[1] Browning G.K.S., *Automatic Typesetting*, University Computing, Vol **6** No 2, pp 95-98, June 1984.

[2] Hamilton-Smith N., Murison J.M. (editors), *EMAS 2900: User's Guide*, ERCC, October 1982.

[3] Hamilton-Smith N. (editor), *EMAS-3: User's Guide*, ERCC, September 1986.

[4] Murison, J.M., *Philips GP300 Printer: Character Sets*, ERCC User Note 50, June 1985.

[5] Hare R.J., *A SCRIBE Primer*, ERCC User Note 66, May 1986.

[6] Shaw S., *Electronic Mail on EMAS*, ERCC, June 1986

[7] Furuta R., Scofield J., Shaw A., *Document Formatting Systems: Survey, Concepts and Issues*, ACM Computing Surveys, Vol **14**, No 3, pp 417-472, September 1982.

[8] Reid. Brian K., SCRIBE *Document Production System - User Manual*, Unilogic Ltd., April 1984.

[9] Unilogic Ltd., SCRIBE *Document Preparation System - Database Administrator's Guide*, April 1985.

Index