MicroScript

Wordprocessing for the CPC464
Soft 1010

Published by AMSOFT, a division of

Amstrad Consumer Electronics plc
Brentwood House
169 Kings Road
Brentwood
Essex

First edition 1985

SOFT 1010

CONTENTS

1.0 Using This Manual

MICROSCRIPT is a powerful aid to office administration with applications ranging from text handling to mathematical calculations.

This manual has been designed with all of these applications in mind. The first part takes you step by step through the system up to the point where you will be able to control all but the most sophisticated routines.

In the later sections, there is a summary of every routine that can be called up on MICROSCRIPT, arranged to provide the most rapid access to this information.

Section headings throughout the manual are set out as complete English sentences and extend over several sections in some cases, to help improve the logical flow of information.

On the first few occasions that the system is used, you should read both the headings and the text.

As you become more familiar with the system, you will find that only the headings need be read; moving into the text only where the continuous headlines do not provide sufficient reminder of the subject matter.

The headings, which are such an important element of this manual, are reproduced as a Contents section, where they provide a compact guide to the operating instructions.

1.1 Throughout this Manual, Command Sequences...

Inputting commands will involve pressing two different types of key.

Several of the keytops are marked with complete words or abbreviations (**[ESC]**, **[CTRL]** or **[TAB]**, for example). These are dedicated to specific functions.

Some commands may require these 'dedicated' keys to be used with the standard alphanumeric keys (**R**, **F**, **?**, *, for example). In this situation, there is no difference between upper and lower case letters.

If a command involves the use of figures, there is the choice between entering through the main keyboard, or using the numeric keypad.

.....are shown in bold type

When a function is to be used as part of a command sequence, it is shown in bold type:

Function 30 means call function 30 (**[ESC]R**‹n›). See the list at the end of this manual.

[ESC] means press the key marked **[ESC]**
[ENTER] means press the key with **[ENTER]** marked on it.

2.0 What is a Word Processor?

Pressures exerted on companies today require increasing efficiency and throughput in the modern office. Recent advances in computing have brought computers and word processing within the reach of small and medium sized offices.

Word processing is the technique which enables computers to handle text often merged with data in a variety of ways. Text of all kinds including letters and reports, can be prepared quickly with a word processing system. A word processor is a typewriter with a difference: it allows you to record text entered through the keyboard and store it on disc before printing. Alterations can be made to a document and any errors eliminated before it is printed.

Word processing can extend into areas such as personalised mail shots and other types of automated text preparation. Using the facilities available on Microscript, you will be able to produce virtually any kind of document in a way which will save time and money.

Microscript has been designed to help those who use word processing in the office and is now available on the CPC464.

2.1 Text is entered through a keyboard...

The keyboard is similar to a conventional typewriter keyboard: it carries the standard alphabetical and numerical keys.

.....on to the screen of the visual display unit...

In many ways similar to a television set, the Visual Display Unit (VDU) displays what you have entered through the keyboard.

A moving block of light indicates the position recognised by the computer. This is called the CURSOR and can take the form of a block of light or an underscore line. If a key is pressed then the character will appear at the cursor position. The cursor then moves one space to the right ready for the next character to be typed in.

.....before being stored on disc

The microcomputer which handles the text processing is mounted in a free-standing 'console' unit. Alongside the keyboard/console are the disc drives: the units which allow information to be recorded on to and later recovered from magnetic storage disc.

2.2 Agreeing on what we mean by Characters...

A character is the most basic unit of text. It is any letter, number, space or punctuation mark which you can enter through the keyboard, and which is intended to appear as part of the 'copy' for output through the printer.

.....and words...

Groups of characters taken together and followed by a space are a text processing word. Not only do grammatical words fall into this category, but groups of characters such as $5000 would also be regarded as words on this definition.

.....and paragraphs

Any block of text which is followed by two **[ENTER]** keystrokes is a paragraph. A paragraph may therefore be as short as a single character, or extend the whole length of a document.

This implies that if you use the **[ENTER]** keystroke to end a line before reaching the right hand margin - as you would when typing an address, for example - you are not creating separate paragraphs.

Not until the address is complete, and an extra **[ENTER]** stroke is needed to force a blank line, will the system recognise the end of a paragraph.

3.0 Starting up the System

3.1 Switch on the mains power supply...

Having checked that the mains lead is plugged in correctly, turn the power switch on the monitor ON, followed by the switch on the CPC464 keyboard unit. So long as power is entering the machine, an indicator light will be illuminated.

.....before loading the system disc...

Even with mains power into the system, the computer is merely a collection of independent units. It needs to be programmed with the instructions which make it function as a word processor.

The operating system program must be loaded from the SYSTEM DISC each time the computer is switched on. Once the computer is ready, the screen will display the following:

‹INSERT COLD BOOT SCREEN›

Insert the system disc into the open slot of Disc Drive A. Slide the disc gently into the drive until it clicks into place. Type:

```
|CPM[ENTER]
```

.....to load the operating system program into the memory

When the disc is correctly in place, the program will be read automatically into the computer's memory in two stages. A light in the door of the disc drive will be on while the program is being loaded.

Once the operating system program is loaded, the screen displays a statement of the form:

```
CP/M 2.2 Amstrad Consumer Electronics plc
Copyright (C) 1984 Digital Research
A>
```

Once the operating system program is loaded, remove the disc from drive A and replace it in its plastic library case.

3.2 Load microscript into disc drive a...

When you receive your copy of MICROSCRIPT, you should make two copies of it, and then also copy the working program files onto all discs that you will be using for wordprocessing if you are using a single drive system. Remember to activate the write protect tag on the original master copy before you start to copy! The original master disc should not be used again, unless the other copies are damaged or erased. One of the two copies should be your security copy and the other the working copy.

Refer to the operator's manual for the computer and its operating system for further details of how to take a copy of a disc.

Insert the working copy of MICROSCRIPT in exactly the same way as the system disc, ensuring that it is loaded into disc drive A.

At the prompt sign

```
A>
```

- type `SCRIPT` in either upper or lower case and press **[ENTER]**.

Documents stored on MICROSCRIPT must also be stored on the disc containing the word processing program. Discs without the program are used as document data discs, and may be used where a system with two disc drives is available. New User Discs must be prepared before they can hold documents. The process, known as FORMATTING, is described in the operating system instructions for your DDI-1.

.....to display the main menu

Whenever MICROSCRIPT is run, a series of options is displayed on the screen. This is the MAIN MENU shown below. Each activity is described in the section indicated:

```
C Create a new document ..... Section 4.0
E Edit or view a document ..... Section 7.0
R Reformat a document ..... Section 9.0
S Search & Replace ..... Section 8.3
P Print a document ..... Section 18.0
F File management ..... Section 19.0
? List index of documents
* Exit to system
```

With MICROSCRIPT in drive A, enter your option.

3.3 Resetting the Computer

When the machine is switched on, it is essential that all internal circuits are reset to a known state. This allows the internal control software to start successsfully.

This action is known as RESET.

4.0 Creating a document with MICROSCRIPT

4.1 Select the create option from the Main Menu

If MICROSCRIPT has just been loaded, insert a formatted user disc into drive B. When the main menu appears, select option C.

Storage space is automatically allocated by the computer and is organised in locations called 'files'. Each of these files must be given a reference name.

4.2 A Document must be named

Each filename consists of three parts:

a disc identifier (of the form A: or B:) - if this is omitted, Microscript will assume that the document is to be created on the current disc drive; a name allocated by the user, containing up to eight characters (including spaces) -JONES001, for example;

an 'extension' of up to three characters, used to indicate the file type (DOC for document, for example). An extension must be shown as part of the filename.

The user-allocated part of the filename may not include any of the following characters:

, < > ; : = ? * [ ] {}

Full stops can only be used to separate the filename and the extension. If a full stop is used as part of the filename, MICROSCRIPT accepts any characters after the full stop as the extension.

With option C selected from the main menu a prompt will appear on the screen:

```
Which Document:
```

Type in a valid filename and press **[ENTER]**. Many installations of MICROSCRIPT will have been configured with the system disc on drive A, and user discs in drive B. To create files on the user disc, therefore, you must provide a name of the following form:

B:*Filename.Ext*

B:JONES001.	LTR 1st letter to Jones
B:AC/1068.INV	Invoice number AC/1068
B:23JUN.REP	Report dated 23 June

Note that files should be named so that their contents are immediately obvious to the operator. It is not a good idea to build up a sequence of files with names such as John1, John2, John3, ...John57 - trying to searchg to find which one of these holds vital information can be very time-consuming.

The rules for filenames would make the following filenames invalid:

`B:AC:1068.INV`	Contains a colon (`:`) in the name
`B:23.06.84.REP`	Full stop in name

After you have created and entered a valid filename the screen will automatically clear and you can begin to type.

5.0 Text format on the Screen

5.1 Text layout is defined by rulers...

As on a conventional typewriter, MICROSCRIPT provides tab settings for the control of 'carriage' (cursor) position. This system enables you to define right and left margins, tab settings and other features which affect the layout of a document.

MICROSCRIPT uses a system of RULERS for this control. A ruler is essentially a line of dots across the screen. Alphabetical characters are placed on this line to define the control points, as in the following example:

```
W.................T.....T...........T.....................R
```

.....which are called into play from memories

MICROSCRIPT has 10 ruler memories, allowing nine different rulers, 1-9 to be stored at any time. The tenth memory is numbered 0 and always contains the ruler being used.

When MICROSCRIPT is run the ruler memories are automatically loaded with predefined rulers. Ruler 1 is loaded into memory 0 when the system is started. The default settings can be changed, but this procedure is not within the scope of a standard MICROSCRIPT document.

There is no limit to the number of rulers which may be called up during the creation or editing of a document. Whenever a new ruler is selected, a copy is automatically transferred to memory 0.

5.2 Up to nine different rulers may be brought into play

A ruler can be recalled at any time through function ·30, as in the following example:

To recall ruler 6, call function 30 (**[ESC]**R), then press **6**
To recall ruler 4, call function 30 (**[ESC]**R), then press **4**

Rulers can be left embedded in the text as they are ignored when a file is printed.

To remove rulers from the screen, call function 12 (**[CTRL]**Y)

5.3 Amend rulers by changing symbols in the ruler line

A ruler can be changed on the screen and then loaded into MICROSCRIPT.

Use one or more of the following 'control' characters to change ruler settings:

Ruler position with NO **[CTRL]** FUNCTION. .....(full stop).

TAB position ..... **T**

RIGHT MARGIN setting ..... **R**

LEFT MARGIN of a block of text whose right margin is to be set 'ragged' ..... **W**

LEFT MARGIN of a block of text whose right margin is to be set right justified ..... **J**

FIRST CHARACTER in a paragraph, where this is to be in a different column from the left margin ..... **I**

CENTRE of a line of text. Text is centred when function 27 (**[ESC]C**) is called (CENTRE TEXT) ..... **C**

CALCULATOR display digit. (A full description of the MICROSCRIPT calculator is given in section 16.) ..... **#**

Make the necessary changes to the ruler: to remove a control character from the ruler overtype the character with a full stop.

Move the cursor to the line above and then down through the ruler to make it active.

To store the ruler in a memory call up function 29 (**[ESC]S**‹n›), and specify the memory number. Once a ruler has been stored it can be used at any time by calling up function 31 (**[ESC]U**‹n›).

5.4 You can copy a line...

By firstly lifting up the text from the line to be copied by placing the cursor at the start of the line, and then calling function 55, (**[COPY]**) `LIFT A LINE OF TEXT`, and placing the line of text in the required section of the document. Move the cursor to the position in the document where you wish it to be placed, then call function 56, (**[ESC][COPY]**) `PLACE A LINE OF TEXT`.

The line of text that has been lifted can be placed down as many times as is necessary. The line of text is stored in the text lift memory until overwritten by another call to lift a line, function 55, (**[COPY]**).

6.0 Correcting errors during Input

6.1 Retype errors noticed immediately

The chances are that you will make a mistake while typing and notice the error as soon as it has occurred. Call function 2 (**[DEL]**) to erase the mistake.

[DEL] is a two part function:

The cursor is moved back one space to be placed over the character to be deleted;

The character directly underneath the new cursor position is erased.

Having deleted the error continue typing the correct version.

6.2 Other types of error must be located...

A proportion of errors made during typing will not be detected until a document is subsequently reviewed or edited. Since changes can only be made where the cursor is located, use a cursor function to move the cursor to the error.

.....by using the cursor functions...

The cursor can be moved to the left, right, up and down.

The functions controlling these cursor movements are 3 - 6: the keys marked with direction arrows.

Move the cursor one place to the LEFT ..... Left arrow

Move the cursor one place to the RIGHT ..... Right arrow

Move the cursor ONE LINE UP from its current column position ..... Up arrow

By holding this key down you can scroll the text back by up to 23 lines. If a bleep is sounded when using this scroll function, this indicates that either the top of the document has been reached or the part of the memory which holds text is full.

Move the cursor ONE LINE DOWN from its current cursor position.....Down arrow

Holding this key will first take the cursor to the bottom of the screen, and then scroll through the text.

Cursor control keys are self-repeating. If they are held down very briefly, their effect will be repeated. Use this facility cautiously, however.

To position the cursor at the bottom of the document, call function 57, go to bottom of document. (**[ESC]B**).

To return the cursor to the top of the document, call function 53 (**[ESC]T**) GO TO TOP OF DOCUMENT,

.....or text movement functions

To move the cursor more rapidly around the screen, the text movement functions can be used; these functions move the cursor by units of text. Using function #14, for example, would advance the cursor to the end of that line.

Use the following functions to move the cursor: Move the cursor right until it reaches the next valid control character on the RULER ..... **[TAB]**

Move the cursor to the first character in the NEXT WORD to the right along the line ..... **[CTRL]T**

Move the cursor to the first character of the PREVIOUS WORD on the line.**[CTRL]R**

Move the cursor to the RIGHT end of the line ..... **[CTRL]→**

Move the cursor to the LEFT end of the line ..... **[CTRL]←**

Move the cursor to the TOP of the screen ..... **[ESC]↑**

Move the cursor to the BOTTOM of the screen ..... **[ESC]↓**

Move the cursor to the TOP LEFT corner of the screen ..... **[CTRL]↑**

6.3 Having moved to the error, make the deletion...

Use an appropriate **[DEL]** function to erase the error.

.....or the insertion as required...

Delete errors before attempting to insert additional text.

As an example, amend the phrase:

```
'Please return this form 'into
'Please return the enclosed form'
```

Handle this modification as a series of steps:

Delete the '`i s`' from the word '`t h i s`' by calling function 2 (**[DEL]**)

Call function 40 (**[ESC]I**) to enter insert mode and type '`e enclosed`'.

Call function 42 (**[ESC][ESC]I**) to exit insert mode.

.....before reformatting the text

During the course of your deletions and insertions, it is likely that unwanted spaces will have appeared in the text. Use the cursor functions to advance to the appropriate point in the text and then call function 9 PULL TEXT to remove the spaces or characters. Alternatively, extra spaces can be inserted by calling function 8 (**[CTRL]N**) PUSH TEXT.

If the text becomes too long for the line length specified on the ruler, words will wrap automatically on to the next line. A word is never broken by MICROSCRIPT.

6.4 Proof read the document...

To check the contents of the document before saving it, return to the beginning, by calling function 53 (**[ESC]T**)

Having reached the beginning of the document scroll through the text using the cursor functions or text movement functions, pausing to make any alterations or amend any errors you detect.

.....before it is saved on disc

When there are no more corrections to be made and the document is to be recorded, call function 52 (**[ESC]E**) EXIT DOCUMENT SAVE.

This function is normally used for leaving the edit mode and returning to the main menu.

During the execution of this function all text is saved on disc and a back-up file is produced on the same disc to correspond to the edit file prior to the last edit session.

6.5 Return a document to disc frequently

While MICROSCRIPT is a proven product, it is always a good idea to save text being entered or edited back to disc at regular intervals. It is recommended that the current document should be saved every two pages, by calling function 52 (**[ESC]E)** EXIT DOCUMENT SAVE.

When a document has been edited, the previous version is automatically retained on disc under the same file name, but with the extension .BAK to indicate that it is a back-up copy. This version of the back-up will be overwritten each time that the 'top copy' of the document is edited, and the modified text written back to disc.

7.0 Editing a document using microscript

Editing covers all the operations involved when a document already recorded on disc is revised.

Having created a document and filed it away on disc, you may wish to check the work before printing. Insertions, deletions and other amendments made at this stage require exactly the same functions as were required during input.

7.1 Recall text to the screen

With the main menu displayed on the screen, select the edit function, E.

The screen will display:

```
Edit function selected
```

But you have to insert the name of the document to be edited, for example:

A:JBK.DOC for a document on drive A
B:020384.LET for a document on drive B

7.2 Locate the cursor using text movement functions...

Since changes can be made only at the cursor position, use one of the many techniques available for moving the cursor to the correct point in the document.

The use of these text movement functions was explained in section 6.

.....or the cursor control functions

The cursor control functions enable you to scroll a document, either vertically through a document or from left to right beyond the width of the screen.

With the cursor at the beginning of the document, scroll through the text pausing to make any errors you detect.

The use of these cursor control functions was explained in section 6.

7.3 Alternatively, search with the 'find' function

Using the FIND function, locate the first occurrence of any word or phrase.

The FIND TEXT function, function 45 (**[ESC]F**), can be called either to look for a piece of text in the current file or to scroll through a file to a known point.

To locate text using function 45 (**[ESC]F**):

Type the word or phrase that is to be found at the extreme left of a blank line. Call function 45 (**[ESC]F**)

MICROSCRIPT searches for the next occurrence which matches the cursor line; deleting the cursor line prior to search.

When MICROSCRIPT has found the search text a keyboard bleep sounds and the cursor moves to the located text.

If no occurrence is found before the end of the edit file, a message appears at the top of the screen. The edit process recommences at the top of the document.

8.0 Making changes to the text you have recorded

8.1 Amendments can be made by inserting new text

When you have moved to the position where the first charcter is to be inserted, call function 40 (**[ESC]I**) ENTER INSERT MODE.

Additional text is accepted by moving all text to the right of the cursor on the current line to make room for new characters.

As an example of the insertion procedure, the phrase 'final draft of the' is to be inserted in the following sentence, immediately ahead of the word 'agenda'...

We are enclosing for your information the agenda for the meeting and would appreciate your comments.

The procedure to be followed would be:

Place the cursor over the FIRST letter, 'a', of agenda.

Call function 40 (**[ESC]I**).

Type the insertions required.

The effect of the insertion would be:

We are enclosing for your information the final draft of the agenda for the meeting and would appreciate your comments.

Call function 41 (**[ESC][ESC]I**), EXIT INSERT MODE.

8.2 Text can be deleted by character...

The delete function, function 2 (**[DEL]**), is normally used for correcting 'sensed' errors, or for making small deletions during editing. (See section 6.1 for details).

.....or by word...

This function is used to delete from the cursor position to the next space enabling either the whole word or part of a word to be deleted.

The DELETE WORD function, function 37 (**[CTRL]V**), operates as follows:

Move the cursor to the first letter of the word to be deleted.

Call function 37 (**[CTRL]V**) DELETE WORD

If the cursor is not placed on a word, a bleep will sound.

.....or by line

An alternative way of deleting an area is to call the WIPE RIGHT or WIPE LEFT functions which erase all text to the right or left of the cursor respectively.

To delete text to the right of the cursor, call function 35 (**[ESC][**), WIPE RIGHT.

To delete text to the left of the cursor, use function 36 (**[ESC]]**), WIPE LEFT.

8.3 Global exchange replaces text throughout a document

If text is only to be located, but not replaced, call function 45 (**[ESC]F**), FIND TEXT. Refer to section 7.3.

The SEARCH and REPLACE feature in the main menu allows one string of characters to be replaced by another. The number of characters does not have to be the same.

SEARCH and REPLACE is accessed from the main menu by selecting the S option.

THE system will request a valid filename for the search. Enter filename then the search string.

MICROSCRIPT provides three options within the search facility:

1.Search and count occurrences
2.Search and delete
3.Search and replace

Option 1 will search for each occurrence of the string within a file and return a count.

Option 2 will remove all occurrences of the search string within the text.

Option 3 will find all occurrences of the string and substitute it with the replace string. If this option is selected, the replacement string must be entered when prompted.

8.4 Additional editing functions are available

Functions which you may wish to use but which have not as yet been covered are:

INSERT (Open) blank LINES in existing text.....13, (**[ESC]O**)

This allows new text to be inserted or the appearance of existing text to be improved.

UPPER CASE 18, (**[ESC]>**) ..... Converts all the characters to the right of the cursor to upper case if the cursor is positioned over a space; otherwise all characters to the next space are converted.

LOWER CASE 19, (**[ESC]<**) ..... Converts all the characters to the right of the cursor to lower case if the cursor is positioned over a space; otherwise all characters to the next spaced are converted.

UNDERLINE the current line ..... 32, (**[ESC]-**)

UNDERSCORE the current line ..... 33, (**[ESC]_**)

At the printing stage, the underscore characters are placed on the same line as the text.

8.5 Edited text is stored on disc...

Calling up a job using the EDIT function transfers a copy of the document on disc into the screen memory. Until you have completed the editing and stored the revised copy on disc, the original document remains unchanged.

.....by saving the document

To retain the edited version of the document, call function 52 (**[ESC]E**), EXIT DOCUMENT SAVE.

Alternatively, leave a document without saving any of the changes that you have made to the edit file. To do this, call function 50 (**[ESC]Q**), QUIT DOCUMENT.

Whenever a document has been edited, a copy is retained on disc under the same user name, but with the extension **.BAK** instead of the extension previously defined.

Subsequent editings of the same file will overwrite **.BAK**; earlier versions of which are automatically deleted.

Example:

> A document is created as **ANGLER.LTR**. After the first editing the unmodified version becomes **ANGLER.BAK**.
>
> On the next occasion **ANGER.LTR** is updated, previous **ANGLER.LTR** is written back to overwrite the unmodified **ANGLER.BAK**.

If you require additional 'layers' of security during editing, a copy of the 'working' document should be taken under a new name (using the file handling procedures described in Section 19 of this manual)

9.0 Reformat a document

This option, R from the main MICROSCRIPT menu should be selected whenever you wish to carry out any of the following functions.

Calculating with MICROSCRIPT
Expanding lines of text (function 44)
Reformatting entire paragraphs of text (function 28)
Cut and Paste (functions 20-26 inclusive)

The 'Reformat a document' option is intended to be used primarily to tidy up a document and enter numeric data, which requires calculation, immediately prior to printing a document.

Calculating with MICROSCRIPT is explained in detail in Section 16.

When you select R from the menu, you will be requested to enter the document filename.

9.1 Expanding a line

The Expand Line function allows all text to be aligned against the current right hand margin until the end of the current paragraph without a reformat operation. Call function 44 (**[ESC]X**), EXPAND LINE.

As an illustration, a paragraph of text is to be output with a flush right hand margin. On input to the screen, it would look like:

```
Office technology has evolved a long way since
the earliest electric typewriters during the
1920's and 1930's. Perhaps the greatest advance
was made immediately after the Second World War
in America by IBM.
```

After calling up function 44, however, the same paragraph would be displayed as:

```
Office technology has evolved a long way since
the earliest electric typewriters during the
1920's and 1930's. Perhaps the greatest advance
was made immediately after the Second World War
in America by IBM.
```

This function will right justify a paragraph of any length from one line to a complete disc file, but it must be called up for each line in turn. (Note that right justification during reformat may only be used on a paragraph shorter than 23 lines, or exceeding one line.)

Should a line containing only one word be encountered during the execution of this function, a bleep will sound and MICROSCRIPT will skip to the next line.

9.2 Reformatting an entire paragraph

Position the cursor in any part of the required paragraph to be reformatted, then call function 28 (**[ESC]J**), REFORMAT PARAGRAPH. MICROSCRIPT will immediately reformat the text in the paragraph according to the margin and indentation markers contained on the current active ruler.

Before you call the functions, you may wish to recall and make active another ruler. This has already been explained in Section 5.

9.3 Cutting and pasting with Microscript

One of the advantages of text processing is that text can be moved around within a document, or between two different files, without the need for retyping.

The term CUTTING relates to marking out a block of text on the screen and transferring a copy of this material to the Cut and Paste memory.

At the Cutting stage, there are three options.

If the block is BLANKED, the text is deleted from the screen and the area is filled with blank spaces, enabling you to insert alternative text at that point.

Alternatively, the text may be LEFT. This function enables you to copy the block but leaves the original text untouched.

Finally, the text may be REMOVED. All text specified in the area is deleted once it has been copied. Subsequent text moves up to fill the space created.

Note that any text remaining on lines from which a block has been removed will be overwritten when the text below it moves up the screen.

PASTING is the opposite of Cutting, and inserts a copy of the text held in memory at another point, either within the same document, or into another file. As with the text lift system, MICROSCRIPT offers three options:

OVERLAY inserts the text at the required point overwriting existing text. This is a 'destructive' operation: any text under the inserted text is lost.

The safest option is to INSERT. A new line is opened for each insertion so no text is lost.

If you wish to move columns of text, use the ELBOW facility. All existing text moves to the right. Note that if the right margin is exceeded, excess characters are lost.

9.4 Sections of text can be cut from one area...

To use MICROSCRIPT's Cut and Paste facility, select the REFORMAT option from the main menu.

To CUT an area of text, the text must be defined by marking the upper left and bottom right hand corners. Move the cursor to the left hand corner of the required text and mark it calling function 20, (**[ESC](**).

Then move the cursor to the bottom right hand corner of the required area and copy the text by calling one of the CUT functions:

Cut and Paste Blank Function 21, (**[ESC])B**)
Cut and Paste Leave Function 22, (**[ESC])L**)
Cut and Paste Remove Function 23, (**[ESC])R**)

Note that the Cut and Paste facility is limited to one screenful.

.....and pasted elsewhere

Pasting extracts a copy of the text held in the copy memory and inserts this material at the cursor position.

To PASTE the contents of the memory, position the cursor at the point where you wish to make the insertion and then use one of the PASTE options:

Cut and Paste Overlay Function 24, (**[ESC]*O**)
Cut and Paste Insert Function 25, (**[ESC]*I**)
Cut and Paste Elbow Function 26, (**[ESC]*E**)

MICROSCRIPT allows you to paste the same piece of text any number of times since the contents of the Cut and Paste memory are only lost during a new Cut and Paste operation or when the current session of Microscript is exited.

10.0 Layout features of microscript

MICROSCRIPT contains a number of functions which can be used to enhance the layout of a piece of text.

10.1 Microscript will completely underline...

When underlining is required, a fresh line is automatically inserted underneath the cursor line and hyphens placed on it from the start to the end of the text present on the cursor line.

To underline, call function 32, (**[ESC]-**)

At the end of the operation the cursor is left at the end of the hyphenated line.

This is underline as shown in print.

.....or underscore

The same principle applies as with the underline except that on printing the underscore characters are placed on the same physical line as the text to produce a true underscore effect.

To underscore call function 33, (**[ESC]**_)

A single semi-colon is automatically positioned at the right hand end of the screen and the underscore characters are placed one line below the text line. The underscores can be edited if specific words only are to be underscored.

This is underscore as shown on the VDU screen ;

```
This is underscore as shown in print.
-------------------------------------
```

IMPORTANT: this is one of the very few areas of Microscript in which the screen representation differs from printed text.

10.2 Multiple strikes of the print head can be set...

There are three techniques which allow the printer head to strike more than once in a single character position:

OVERSTRIKE - in which two successive lines of text are printed in one line space.

EMPHASISED - in which the character is struck twice in exactly the same position.

EMBOLDENING - in which a character is struck twice, with the head displaced by a very small amount between the strokes;

Although the commands to overstrike, emphasise and embolden are displayed on the screen, the effect is only seen at the printing stage.

.....to print two lines of text in a single line space...

A semi-colon in the last column position (usually column 79) prevents the printer from advancing after to the next line.

```
000000000              ;
---------
```

At the printer stage, the result would be:

00000000

.....or text which is either emphasised...

MICROSCRIPT recognises semi-colons in the last two columns on a line as an instruction to strike each character twice for emphasis.

Firstly, you must use the underscore function, 33, (**[ESC]**_), and then add an extra **;** immediately to the left of the one already on the line. The underscore will not be printed.

```
CHAPTER 1: The origins of the word processor      ;;
```

With two semi-colons the effect would be:

CHAPTER 1: The origins of the word processor

.....or emboldened

The procedure for emboldening (bold printing) is similar to that for the double-strike, with a third semi-colon placed one column further to the left.

As an example of the technique.

```
CHAPTER 1: The origins of the word processor        ;;;
```

Three semi-colons produce emboldening:

```
CHAPTER 1: The origins of the word processor
```

11.0 Formatting Features

A word processor produces the highest quality of display with the minimum effort at the keyboard. Automatic title centering, justification and tabulation are some of the features which are provided for this purpose.

11.1 Text may be centered automatically...

Any line of text can be centered between the current left and right hand margins by calling function 27 (**[ESC]C**), CENTRE LINE.

This function will re-write the contents of the cursor line centrally under the control letter **C** in the current ruler.

11.2 Tab settings and indents aid formatting

Tab settings were outlined in section 5. They can be used to provide several types of indentation for text, as in the following examples:

```
(a) ....J....I.....................R.....
```

This illustrates the effect of using an indent code (I) in the ruler to offset the first line of text in a paragraph. The operator has to use the TAB key or cursor arrows to move to the position shown by the letter I.

```
(b) .....I....J.........................R.....
```

This is the reverse of the previous example, and holds the start of the first line out from the body of the paragraph.

If the I code were not used in either example - there would be a problem created when the paragraph was re-justified.

```
(c) .....J....T....T.........T.........R.....
```

There are situations when text and data has to be set out in neat columns: the tab setting (T) comes into its own, as in the example:

```
1 Aluminium  16.09  grams
2 Magnesium  17.98  grams
3 Tungsten   25.97  grams
```

11.3 Microscript will wrap text around line endings

When the end of a line is reached, MICROSCRIPT automatically 'wraps around' to the margin of the next line.

The following sequence of illustrations show how MICROSCRIPT handles the formatting of text at line endings:

```
.....J.........................R.............
```

Words and figures are accepted onto

```
.....J.........................R.............
```

Words and figures are accepted onto a sing

```
.....J.........................R.............
```

Words and figures are accepted onto a
single line of text until there is no more s

```
.....J.........................R.............
```

Words and figures are accepted onto a
single line of text until there is no more
space available. Words are not broken.

12.0 Operating microscript in different modes

MICROSCRIPT provides a choice of seven different MODES of operation.

To operate MICROSCRIPT in another mode, it must be in EDIT mode, with the system controlled either through the keyboard or a programmed sequence in a control job. (Programming is described in section 22.)

These operating modes are designed to provide flexibility: the system can be switched between modes whenever required.

MICROSCRIPT automatically cancels all alternate modes on returning to the main menu, unless there are specific commands in force which would prevent this.

To enter an alternate mode call function 48 (**[ESC]M**‹n›), ENTER MODE 'n'.

0 VERTICAL TEXT

This mode changes text entry to a vertical downward cursor movement, so that each character typed appears on the next line immediately below the previous entry.

1 AUTO TAB

Pressing [ENTER] in mode 1 causes the cursor to indent to the left margin on the next line, rather than the left hand edge of the screen. This facility is useful when large amounts of indented text have to be prepared.

2 HOLD LINE FORMAT DURING MERGE

When text is merged from another document or a memory, it formats automatically to the margin settings in force at the point of the merge in the current document.

Mode 6 overrides this condition and preserves the line format of the incoming text intact.

3 DEFEAT ALL VISIBLE EQUIVALENTS

This mode prevents the visible equivalents of control characters from being interpreted on the screen. This programming facility allows visible equivalents to be stored in abbreviation memories and inserted into a document without the functions they represent being made apparent at that stage.

As an example, the symbol ~ would normally be interpreted as a [ENTER] keystroke when it was merged into the current document from memory. If the system is held in Mode 3, the symbol will be merged in its 'equivalent' form.

4 INVISIBLE TYPE

Mode 4 is intended as an aid to the system designer. All write to screen operations are disabled. This allows an operator to program the system to handle such functions as line deletions and screen clearing in the 'background'. Programs written in Microscript take on a more professional appearance to the end-user.

Mode 4 is intended as an aid to the system designer. All write to screen operations are disabled. This allows an operator to program the system to handle such functions as line deletions and screen clearing in the 'background'. Programs written in Microscript take on a more professional appearance to the end-user.

An example of how the mode should be used would be a routine for preparing invoices. The cursor is suspended while text and data are manipulated around the screen.

5 INSERT MODE

MICROSCRIPT is set by default so that any characters entered at the point of the cursor will overwrite any characters on the screen in that position.

Mode 5 provides the alternative method of entry, in which new text will automatically force existing text to the right. Material held on the screen can only be removed using one or more of the Delete Text functions.

6 HOLD SPACE FORMAT DURING MERGE

When text is merged from another document or a memory, it formats automatically to the margin settings in force at the point of the merge in the current document.

Mode 6 overrides this condition and preserves the format of the incoming text intact.

7 PROGRAM LEARN

This mode is invaluable to a MICROSCRIPT user developing programs. The system will automatically convert function sequences entered through the keyboard into their symbolic form for loading into the current document.

With the system in Mode 7, pressing the **[ENTER]** key would display the sequence ! rather than move the cursor to the next line of the screen.

The use of Learn Mode (7) is discussed at length in Section 22 of this manual.

To exit from an alternate mode call function 49, (**[ESC][ESC]M‹n›**) EXIT MODE **'n'.**

13.0 The merge text interpreter

Using the facilities described up to this point, a document can be prepared and edited.

There are occasions, however, when a document is to be compiled from material which is already held on disc. MICROSCRIPT includes two methods by which text already held on disc can be merged into a document being edited.

The recall of a merge file.....Section 14.0
The recall of a textual abbreviation.....Section 15.0

In both cases text is passed through a control system known as an INTERPRETER, which ensures that when text is brought into a document, unnecessary spaces are removed to preserve the format.

If MICROSCRIPT is being used merely as a word processor, this will be the extent to which the interpreter is used.

The interpreter has a second and more important function however, understanding and executing control symbols. This ability transforms MICROSCRIPT from a word processor into a powerful system capable of complex, pre-defined operations.

In this way, MICROSCRIPT can be programmed to perform many tasks including data manipulation, so that complete systems can be built to perform tasks including invoicing and data retrieval.

13.1 Use the interpreter...

The interpreter may be used to improve the loading of text from a merge file such as a standard paragraph into a document through the insertion of two 'return' sequences at the end of one paragraph to force a blank line prior to the start of the next.

Control charaters may be inserted anywhere in the text and will function as though the key sequence had operated manually at that point.

.....to execute codes embedded in a document

The interpreter defines the control characters that can be used in a merge sequence.

A VISIBLE EQUIVALENT consists of a 'marker' character which is rarely used in other areas of text processing. The character which immediately follows the marker represents the function required.

In a system being used in an English-speaking country, the tilde symbol (~) makes an ideal choice for the marker.

14.0 Merging Files

MICROSCRIPT allows the document being edited to be merged with any disc file.

To merge files call function 39, (**[ESC]G**) CALL MERGE FILE.

The text present on the cursor line is taken as the file specification. Type the file specification at the start of a free line and then enter the control sequence. The file specification is automatically removed from the document.

If the file called does not exist then the system bleeps and the cursor returns to the start of the line.

All files merged using this method pass through the interpreter (Section 13). Each character is read, and any control codes present are executed.

Merge files are read from start to finish using MICROSCRIPT. Alternatively, the disc can be indexed to enable a simple disc file to contain separate sections, each of which can be individually read.

14.1 Generating text

The simplest form a merge file can take is a standard paragraph. But a complete libary could be built up containing standard paragraphs to enable rapid text generation.

A control sequence such as 'Use ruler 5' could be inserted at the start of the paragraph to ensure that a particular ruler is used for the recall of that piece of text. The paragraph could then be ended with a 'reformat' to complete the text recall in a predictable format.

During text merge, the symbol ! is used to halt the system for an input. When MICROSCRIPT finds this character in the text being merged, the merge process pauses temporarily: text and data can be input from the keyboard. MICROSCRIPT will load this material into the document. To continue the merge press **[ENTER]**. To abandon the merge process at any stage press **[ESC]**.

14.2 Index merge files...

MICROSCRIPT offers a simple method of indexing which allows more than one paragraph or section to be held in a single disc file and still be accessed individually.

Each separate section is given a (number) key to identify it from the next and is ended by the sequence)).

When a disc file is specified, this (number) key can be entered to qualify the filename.

MICROSCRIPT searches the whole file until the key is found and merges until the end marker is found. The end marker can occur on a new line or be placed after the last character.

The call procedure for an indexed section is similar to the usual call sequence except that the disc filename is followed by a comma and then the key to be sought:

(filename.ext),key (function 39 **[ESC]G**)

.....so that microscript can switch between parts of a file

You can program a merge file to jump automatically to a different part of the disc file during merge. This instruction could be used to structure a standard letter system on one disc file. It follows that standard letters could be programmed to jump to a common ending such as 'Yours sincerely'.

The technique provided for 'jumping' also uses the principle of an indexing key within the merge file. It is described more completely in Section 22 (Programming).

MICROSCRIPT searches for the key from the top of the disc file. Thus the jump instruction may be used to cause either a jump forward or a jump back effect.

14.3 Nesting merge files

You can call merge files through a command embedded in another file. When function 39 CALL MERGE FILE is encountered within a file already being merged, the system will call up the new merge file and bring this to the current document before proceeding with the original merging process.

Since MICROSCRIPT provides for up to seven levels of 'nesting' of merge files, a complex hierarchy of files can be assembled for an application.

15.0 Manipulating text through memories

The concept of a temporary memory was introduced in section 9, (Cut and Paste operations). A further ten memories are provided for text abbreviation, which may contain text, command sequences or a mixture of the two.

The storage capacity of these memories is limited to one screen line less two spaces but you can store extra text if one abbreviation contains the commands required to call another.

Each abbreviation memory has its contents preset when the system is first generated. Changes may be made, however, to any or all of the memories for the duration of an editing session.

Preset abbreviations are used for commonly used phrases or control sequences and they will remain the same each time the system is run.

To call a preset abbreviation:

Call function 47, (**[ESC]**@n) RECALL ABBREVIATION 'n'

To store text or a command in one of these memories:

Call function 46, (**[ESC]**#n) STORE ABBREVIATION 'n'

If a memory is empty, the contents of the cursor line will be loaded. Where there is already information in a specific memory, however, this will be displayed on the screen with the memory number.

At that stage, you have to indicate what is to be done with the data already held in the memory you have identified. Press one of two characters:

C To clear both the temporary information from the screen, and the memory contents;

L To clear the temporary information on the screen, and leave the memory contents unchanged.

16.0 Calculating with MICROSCRIPT

A major feature of MICROSCRIPT is a fully programmable, five function calculator which enables you to total columns vertically and horizontally, use ten numeric memories and full decimal point justification.

16.1 Define the Limits of the Calculator Fields

Rulers are used to specify the exact location of the calculator. The characters '·' and '#' are placed on the ruler to define each display position.

Typical calculator displays would be:

```
.........#####.................#####........##########......
.............#####.##.....#####.##.....#######.##..........
.............................###.##...###.##...####.##....
```

You may use any number of displays on a ruler subject to the ruler length. Any one display may be up to fourteen numeric places with no restrictions on the position of the decimal point.

16.2 Make Calculator entries...

To enter calculator mode, select REFORMAT from the main menu. Ensure that the active ruler contains the appropriate marker for an arithmetic field. Note that whilst in calculator mode all cursor control and general control sequence operations are invalid.

Use the **[TAB]** key to advance the cursor to the first arithmetic field: the cursor remains one column to the left of the field, and three zeroes will appear within the field:

```
.............................###.##...###.##...####.##....
                              ■  0.00
```

Enter the figures required, using the **decimal point** (.) to jump from the integer block (whole numbers) to the decimal places (if a decimal place has been defined):

The value 34.09, for example, would appear as:

```
.................................###.##...###.##...####.##....
                                 ■ 34.09
```

At this point, you have the choice of leaving the current arithmetic field, or performing a calculation on the value entered there:

Press **[TAB]** to move the cursor immediately to the right of the field - a non-numeric area.

or Alternatively, use one of five functions to perform a calculation:

Plus	+
Minus	-
Multiply	*
Divide	/
Percent	%

Immediately any of these functions is entered, the field clears to await the next value.

...using logical sequences...

The sequence of instruction is similar to that of most scientific calculators: you enter the sequence 'as you would say it', for example:

```
9*33-12= is a valid sequence
```

This would calculation would be entered on the display as:

```
.................................####.....................
                                 ■   9

.................................####.....................
                                 ■   * (Does not appear on screen)

.................................####.....................
                                 ■  33

.................................####.....................
                                 ■   - (Does not appear on screen)

.................................####.....................
                                 ■ 297 (Product of 9 x 27)

.................................####.....................
                                 ■  12
```

```
.....................................####.....................
                                     ■      = (Does not appear on screen)

.....................................####.....................
                                     ■ 285
```

The valid sequence to calculate 12% of 45 would be....

```
45 * 12% =
```

Pressing a non-numeric key causes the keyboard to bleep and zero is entered in the display field.

...being careful not to overfill the field

If you attempt to overfill a display field, the calculator will automatically overrange. This is indicated by a keyboard bleep followed by a display field filled with ! characters.

Attempting to enter the value 61234 into the arithmetic field used in the last example, would give:

```
.....................................####.....................
                                     !!!!
```

16.3 Manipulate data in a numeric field...

In addition to the five arithmetic functions supported by Microscript, there are a further six arithmetic routines. The first four relate to the contents of specific field::

S n	Store a value in numeric register n (0 to 9)
R n	Recall the value in numeric register n (0 - 9)
X	Switch the sign of a value from positive to negative
C	Clear the field immediately to the right of the cursor

Note that numeric registers are temporary and will loseall their contents when the system is taken out of edit mode.

...and calculate the horizontal total...

When more than one field has been set out across the screen page, the total of all the data in the fields on a single line can be calculated.

Place the cursor immediately to the LEFT of the field in which the total is to appear, and press **H**.

```
.................####.##.....####.##.....####.##............
                  123.45       234.56      ■
```

Then press **H**...

```
.................####.##.....####.##.....####.##............
                  123.45       234.56      ■ 358.01
```

Note that if the calculator program finds a non-numeric character in a field, the horizontal totalling cannot proceed any further to the left margin. In the following example, the letter E in the second field would prevent the value in the first field from being calculated into the total:

```
....####.##.....####.##.....####.##.....####.##............
    1000.00         E         1000.00     ■1000.00
```

...and the vertical total

Where a column of values has been entered into an arithmetic field, its total can be found by locating the cursor immediately to the left of the field where the total is to appear, and pressing **V**.

The arithmetic program will calculate the total of all the values in the same vertical column of fields. It is possible to hold sub-totals by filling one screen line in the filed with hyphens. MICROSCRIPT will only total up to this line and no further.

16.4 Exit from the calculator

To leave calculator mode, tab out by pressing **[TAB]**. The cursor moves one space to the right of the display field.

If the final display field contains a value other than zero, then the result is left as text on the document. Otherwise the display field is left as blank spaces.

If you wish to leave the calculator during an overrange condition, then the display field is blanked as though a zero condition existed.

NB: Pressing **[TAB]** also allows you to move across columns within the calculator.

17.0 Preparing a document for printing

The routines to be followed for printing a MICROSCRIPT document are described in Section 18 of this manual.

When a document is being created or edited, however, it is possible to embed a series of printing commands into the text. These codes are read by the computer and interpreted only when the document is sent down from the console to the printer unit.

There are three types of command which can be recognised in this way:

17.1 Enter bracketed commands on lines without normal text

Bracketed commands consist of a single character enclosed by brackets.

A bracketed command is placed at the far left of the screen. Once the system has identified the command, any text on the same line will be disregarded.

MICROSCRIPT recognises the following commands:

<F> To terminate printing on the current sheet and force a page feed on the printer. Page numbers will be printed if appropriate and all other controls and sequences will recognise the page break.

<R> To indicate a remark line. This indicates to MICROSCRIPT that any text on the current print file line should be ignored: there is no other command implied.

<S> To indicate that the next line is a sub-title and must not be printed closer to the bottom of the page than the number of lines specified in sub-title minimum. (See Section 18.2)

<T> To indicate the current line as the title line for each page to be printed. This may be changed at any time by using this command.

<I> To identify the text on the rest of the current line as the definition for an inclusion file. Printing of the current file is suspended until the inclusion file has been printed: it then resumes. An inclusion file is identified in the same way as a merge file during editing.

```
<I>B:NEXTFILE.PRT,#A
```

<W> To mark the start of wide print mode. (See Section 17.2)

<X> To mark the end of wide print mode. (See Section 17.2)

17.2 MICROSCRIPT will print in wide mode

In order to print documents and reports requiring a greater line length than can be displayed, MICROSCRIPT will operate in wide mode.

This mode will permit line lengths of up to 255 characters for a maximum of 100 printed lines.

Sections are built up and added to the end of the previous one building the line internally. It follows that if the last character of one section is in column 58, for example, the first character of the next section will be printed in column 59 unless at least one **[ENTER]** stroke had been placed between the sections.

Mark the start of the section using the bracket command <W>. This line should not contain any text - only this command - and a marker - > -. This corresponds to the end of text lines in this section.

Mark the end of the section with <X>. MICROSCRIPT will then return to normal mode.

The sections must contain the same number of lines even if this involves inserting blank lines.

18.0 Printing text with MICROSCRIPT

Printing text from MICROSCRIPT is a separate option from `Create` or `Edit`. The routine for printing is accessed through the main menu.

```
C   Create a new document
E   Edit or view a document
R   Reformat a document
S   Search & Replace
P   Print a document
F   File management
?   List index of documents
*   Exit to system
```

Selecting option `P` from the main menu will display the following screen messages after a short pause:

```
Press ENTER to set print parameters or SPACE to feed paper


Enter Document Name, m?_ to list documents or ESC to exit
```

Select `?` to generate a list of documents; specifying the drive correctly (A or B) when the prompt appears to request the drive name.

18.1 Print a document directly by entering its name

In the majority of MICROSCRIPT applications, a document will be printed directly from disk, without any changes being made to the default format details, or any formatting codes which may have been embedded in the text to produce 'local' printing effects.

If the required document is in drive B, remember to enter `B:` in front of the document name. To print the document '`TEST1`' currently held on the disk in drive B, for example, the response to the prompt would be:

```
B:TEST1
```

MICROSCRIPT will select the document from disk and prepare to print it.

During the print run MICROSCRIPT will display the current page and line numbers on the screen together with full details of the files being printed.

18.2 Alternatively, change one or more print parameters...

When MICROSCRIPT is loaded from disk, the system contains a number of default printing 'parameters' - settings which determine features such as the number of copies to be printed, and the position of page numbers on each sheet of output.

Pressing **[ENTER]** when the initial screen prompts appear displays a menu of print parameters. Check the list and press **[ENTER]** to accept all 18 as they stand.

1	Page size	Defines the size of each sheet of paper in printed lines. The value entered here is used to calculate the page numbers. Minimum value is 10.
2	Line to end text	Defines the line at which printing will end. Paper is spooled through to the start of the next page. Minimum value is 10.
3	Left start point	Used to offset the left hand printed edge. It enables printing to be set up to correspond with sheet feeders.
4	Pause at top	Permits a pause at the top of each physical page if required.
5	Sub-title minimum	Prevents the sub-title from being printed unless a specified number of lines remain.
6	Double space	Used to switch on or off the double line space facility.
7	Start page	Defines the point at which printing is to start.
8	End page	Defines the point after which printing is to stop.
9	Page no. style	Defines how the page numbers are printed. The caret symbol (↑) represents the page number, so the entry - ↑ - would indicate a number of the form - 123 -.

10	Position of nos.	Contains a description of the position on a page of the printed page numbers, for example, A or A2 results in alternate page numbering.
11	Line no. for page nos.	Defines the line number at which the page number will appear. Ø switches the page numbering facility off.
12	Title position	Controls the position of the title. Valid entries are L (left), C (centre), and R (right).
13	Line no. for title	Defines the line on which the title will appear. Ø switches the title facility off.
14	Line length	Defines the width in column units of the paper being used.
15	No. of copies	Usually set to 1 but can be altered to rerun the print program.
16	Line to start text	Defines the line number on each page at which printing will start.
17	No. of first page	Allows you to offset the start number for the first page of the file. By setting the parameter to zero you can leave the pages unnumbered.
18	User option each page	Allows you to send one of the preset printer controls to the printer before the start of each page.

To make a change to any one or more parameters, **for the duration of the current printing session:**

Enter the number shown against the required parameter
Press **[ENTER]**
Enter the new value for the parameter
Press **[ENTER]**

When all the changes have been made, press **[ENTER]** once more, to display the original screen prompts for printing.

```
Press [ENTER] to set print parameters or SPACE to feed paper

Enter Document Name,m?_ to list documents or ESC to exit
```

...before starting to print a document

Once all the parameters have been set, advance the system to file print by pressing **[ENTER]**.

Enter the filename when prompted and press **[ENTER]**. MICROSCRIPT will search for the file and then pause to indicate that print run can begin.

During the print run MICROSCRIPT will display the current page and line numbers on the screen together with full details of the files being printed.

The print run can be halted at any stage by pressing **[ESC]**. The system will then return to edit mode.

19.0 Use file management to organise a disk

MICROSCRIPT contains a file management system for the movement and organisation of documents on a disk. Select the file management system from the main menu by pressing:

F [ENTER]

The screen will prompt:

```
File Management selected
```

and the following sub menu is displayed:

```
1. Re-name
2. Erase
3. Copy
4. Merge
5. List of documents on drive
```

19.1 Documents may be renamed...

By selecting option 1, a file can be renamed.

The screen prompts:

```
Re-name...Which Document
```

Enter the name and **[ENTER]**.

...or erased

Select option 2 from the sub menu to erase a file.

The screen prompts:

```
Erase...Which Document
```

Enter the name and **[ENTER]**.

With the name entered, the screen prompts:

```
Press [ENTER] to exit or * erase
```

Pressing **[ENTER]** causes the sub menu to appear and the file selected for deletion is not erased.

Pressing * followed by **[ENTER]** deletes the file.

Note that once a file is deleted, there is no way to recover it unless a back up disk has been made.

19.2 Copy files to other disks...

This option is intended for copying single text and data files from one disk to another. It will NOT copy program files or complete disks as back-up copies.

Remove the MICROSCRIPT disk from drive A and replace it with the disk which holds the file to be copied.

Insert a formatted disk into drive B. Select option 3.

The screen prompts:

```
Copy-Enter Filename...
```

Enter the filename and press **[ENTER]**. The screen then prompts:

```
Copy-Enter Filename...
```

Enter the filename and press **[ENTER]** - this is often the same name as the original document, but remember to use the prefix `B:` if it is to be copied onto the disk in the other drive.

Copying is then carried out. Once the process has finished, the menu reappears.

...or merge two files

By selecting option 4 from the file management menu you can combine two files together to produce a new file.

With option 4 selected, remove the MICROSCRIPT disk from drive A and insert the file disk in its place. The screen prompts:

```
Merge...Which Document: #1
```

Enter the filename and press **[ENTER]**.

The screen then prompts:

```
Merge...Which Document: #2
```

Enter the name and press **[ENTER]**.

The final prompt appears:

```
Which Document...Merge -2
```

Enter the name of the merged file and press **[ENTER]**.

19.3 List the documents on a disk

You can view any of the documents on a particular file by selecting this option.

With option 5 selected, the screen prompts:

```
Which driveA: orB:
```

Enter the drive number and press **[ENTER]**.

The screen then prompts:

```
Document name Extension or Return for all on
the disk Pressing [ENTER] will cause all the
files stored on that disk to be listed.
```

An alternative way to list the documents on a disk is to select the appropriate option from the main menu.

With the main menu on the screen press:

?

The procedure is then the same as for option 5 in the File Management system.

20.0 The MICROSCRIPT system software

20.1 MICROSCRIPT is supplied as files on a disk

A standard factory issue software disk is supplied with a series of files.

FUNCTION	FILE
Master editor	SCRIPT .COM
Print system	SCRIPT-P .COM
Search and Replace	SCRIPT-S .COM
File management	SCRIPT-F .COM
Standard command file	SCRIPT .SIF
Reformat	SCRIPT-R .COM

There is also a back-up copy of the `SCRIPT.SIF` file (with the same `.SIF` extension).

21.0 The system information file

The System Information File (SIF) is a disk file containing the text messages and configuration details of the MICROSCRIPT program. On every occasion that MICROSCRIPT is run, the program makes reference to the SIF file; extracting details such as the text of screen prompts and function call sequences from the list of information that SIF contains.

22.0 Programming Applications in MICROSCRIPT

Sections 1 to 21 of the user guide have considered the role of Microscript as a word processing system. In practice, the Microscript software provides a comprehensive range of facilities for developing business application programs.

22.1 The merge interpreter acts on program instructions...

The key to the programming capabilities of Microscript is a piece of software known as the MERGE INTERPRETER. This feature was outlined briefly in Section 13, when the procedures for merging standard paragraphs and abbreviations were explained.

In the context of applications programming, the merge interpreter reads sequences of program instructions and converts these to actions either in the computer's memory or on the screen. The interpreter responds to these instructions as though they had been entered directly from the keyboard. A command to delete the current line, for example, would be treated as though Function 12 (**[CTRL]Y**) had been entered. It follows that for every one of the standard Microscript functions, there is a corresponding command sequence for use in programming.

...which are represented by visible equivalents

Microscript functions such as 'delete word' and 'underscore' are normally actioned immediately they are entered through the keyboard. A method is required, therefore, for entering instructions as 'static' commands which will not be acted upon until the program of which they form a part has been passed through the merge interpeter.

That method is the 'visible equivalent' - a shorthand notation which can be stored in a program to represent the action which will later be executed.

Equivalents for some frequently used functions are:

~**D** for **Function 36** - wipe left

~**B** for **Function 19** - convert to lower case

~**!** for **Function 7** - move cursor to next tab

~**P** for Function **48**, followed by **3** - switch to alternate mode 3

Note that all equivalents are preceded by a special symbol - in this series of examples, it is a tilde (~). This indicates to the merge interpreter that the next character - and the numeral which follows it if the command requires one as part of its syntax - is to be interpreted as an 'action', rather than pure text, when the program is run.

Strings of commands may be executed (interpreted) in sequence: each command being preceded by the special symbol for visible equivalents. The application might require that the cursor is advanced two tab positions to the right, for example, followed by a deletion of all the text on the current line to the left of the cursor. The appropriate sequence would be ~'~'~D. The tilde is the default symbol for visible equivalents because it is so rarely used for any other purpose in the English language.

There is a facility, however, for defeating the effect of a single visible equivalent, making it possible for even the 'special~ character to be used in text. The sequence ~\ will suppress the next sequence that would otherwise be interpreted as a command: ~\~D, for example, would allow the sequence D to pass through the interpreter as pure text.

Note that the suppression only applies to the character immediately following the 'defeat' expression. There are situations - explained fully in section 22.3 - where a long string of equivalents are not to be interpreted until a later stage. Alternative mode 3 will suspend the effect of the equivalents until that mode has been cancelled.

22.2 MICROSCRIPT can generate equivalents automatically

The nature of a visible equivalent is such that it often bears no apparent relationship to the action it represents. It is not immediately obvious, for example, that ~! is the expression for ([**ENTER**]), or that the instruction to move the cursor one line down the screen is ~&.

A user developing an applications program under Microscript would find it time-consuming if the equivalents had to be looked up in a table, and then keyed into the program. Alternative mode 7 of Microscript avoids this by taking each keystroke and automatically displaying the visible equivalent.

With mode 7 activated (Function 48), pressing the ([**ENTER**]) key generates ~! on the screen, while moving the cursor down by one line displays ~& without further action on the part of the user.

To switch mode 7 off, call up function 49 and press 7.

22.3 Programs may be paused to accept keyboard input

A command may be embedded in the sequence being passed through the interpreter to make the system pause for a keyboard input. The visible equivalent is ! Data entry is only possible with the cursor at the left hand edge of the screen.

When the system recognises the instruction, control of the program returns to the operator. Any string of text or data may be entered at that point; the input being terminated by pressing (**[ENTER]**). The program then moves on to the next command.

Alternatively, the **[ESC]** key may be pressed, for the program to be abandoned, and the system returned to the Microscript menu.

Example

A short program is required, to request the name of a Microscript document, then store this response as a temporary text abbreviation in memory 6.

The steps required are:

Provide a screen prompt	'Please enter filename'
Return cursor to left edge of screen	**Function 15**
Pause to allow input from keyboard	!
Clear line to right of cursor	**Function 35**
Store filename as abbreviation 6 and replace existing contents.	**Function** 46 followed by 6 and R when requested.

The visible equivalents recorded in the program would be:

Please enter filename~/!~C~N4

22.4 Programs written under MICROSCRIPT may be stored...

There are two ways of storing a Micro script applications program so that it can be recalled for use when required. The method selected will depend on the nature of the application, and the size of the program involved.

...as temporary abbreviations...

It may be appropriate to amend the contents of one or more of the abbreviation memories for the duration of the current input/editing session. The procedure for recording a program this way is the same a that for storing any character string as a temporary abbreviation:

(program string)Function 46 followed by **n**
where **n** is the number of the the register.

There is a limit of one line length less two columns (normally 78 characters) for the size of a program that can be stored in one of the abbreviation memories.

Whenever an abbreviation memory is called up, any visible equivalent stored there will be 'translated' as it passes through the merge interpreter. This would create problems if the memory were being used to hold repetitive strings of commands for assembly into a larger program still under development. The code for alternative Mode 3 (Function 48 followed by 3) could be inserted at the beginning of the abbreviation memory (and switched off at the end), to allow the string of visible equivalents to pass through the interpreter without being converted to actions at that stage.

Example

A program is being written, which has to call the same sequence of commands several times. Load that sequence - shown below - into abbreviation memory 4 in such a way that it can be used at any time while the program is being assembled (the sequence is not to be run immediately):

```
Goto top left of screen

Use ruler 6

Activate Ruler 6

Advance to second tab defined by ruler 6
```

Using the 'learn' mode of Microscript, the appropriate 'program' would be entered as:

[ESC]M followed by **7**	Enter Alternate Mode 7 (learning on)
[ESC]M followed by **3**	Defeat all visible equivalents
[CTRL] ↑ up arrow	Goto top left of screen.
[ESC]U followed by **6**	Use ruler 6
Function 1	Activate ruler 6
[TAB] (twice)	Move to second tab on
[ESC][ESC]M followed by **3**	Restore effect of visible equivalents
[ESC]#4	Store all characters to the left of the cursor in abbreviation 4.
[ESC][ESC]M followed by **7**	Leave alternate mode 7 (learning off)

On screen, the same sequence would be shown as:

```
P3~F~?6~!~'~'~Q3~N4
```

Note that switching to mode 7, and returning to normal screen entry, are transparent to the program and the user, and no code appears on the screen to denote this change.

...or as files on disk

Programs held in abbreviation memories are restricted in size (even linking all ten memories together gives a total capacity of only about 750 keystrokes).

There is no constraint, however, when programs are stored as files on disk (subject only to the space available there). Any visible equivalents contained in a program file will be actioned when that file is drawn into the document on screen through the merge interpreter (unless, of course, the effect of an equivalent has been defeated).

Microscript disk files may be nested together, so that a 'merge file' command in one file may call up another named file. A maximum of seven 'layers' of nesting is permitted.

22.5 Disk files may be indexed to increase their flexibility

Without any instruction being given to the contrary, the whole of a named file will be called up from disk and merged into the current document. The following command, for examle, would merge `INPUT.DOC` completely:

`INPUT.DOC` followed by **[ESC]G** (Call Merge File)

It follows that ten blocks of instructions held on disk would require ten separate disk files (documents), each with a unique name to identify them. But it was demonstrated in Section 14. that specific parts of a named file may be merged into the current document.

That procedure involved INDEXING disk files in such a way that segments of a single file could be extracted as required, and passed through the merge interpreter. Microscript applications programs can make use of the same facility.

Files may be broken down into units of any length, using a suitable identifier (the usual term for this is 'label'). The label, and the)) which marks the end of such a program unit, are each placed on a separate line.

The document `INPUT.DOC`, for example, might contain three such program units, labelled `START`,`TEST` and `FINISH`. Calling the second of these would involve the sequence:

`INPUT.DOC,TEST` followed by **[ESC]G** (Call Merge File)

Note that the command to pass `TEST` through the interpreter came from outside the document file `INPUT.DOC` which contained it - in this case, the instruction was provided by **[ESC]G** appearing in this current 'working' document open on the system.

22.6 The 'CALL MERGE FILE' command aids programmability...

Extremely sophisticated programming is possible under Microscript if menus of options are built up at the front of an application. **[ESC]G** (`CALL MERGE FILE`) is used to chain from the menu into specified programs, as shown in Section 21.4.

A series of accounting routines might be appended to the main menu of MICROSCRIPT, for example:

...and allows routes to be set up to default programs

The 'call paragraph' command is used to considerable effect in menus such as the one displayed above. Its scope can be increased still further, however, if the command is provided with one or more default 'routes' if the first, and subsequent programs, cannot be located for any reason.

Consider the instruction:

```
INVCE~G DAYBK~G ACMEN~G
```

The program names are read in sequence from left to right. If `INVCE` is not present in the system, the system defaults immediately to the next valid program name - in this case `DAYBK`. And if this program cannot be found, the system is routed through to the third name - `ACMEN`.

While the availability of defaults and second defaults might have only academic relevance in systems based on a simple menu structure, they take on greater importance when the user's response is taken by the system as part of a program name.

A system might be controlled through a menu of perhaps three programs:

```
A   Accounting package
W   Wordprocessing
S   Spreadsheet
X   Exit from system

Which system do you want?
```

Within the system, the programs might be identified as `ACCTA`, `ACCTW`, `ACCTS` and `ACCTX` respectively.

The Call Paragraph function could be triggered by the response to the screen prompt:

`ACCT`**(Pause for input)**`~G`

Entering `A`, `W`, `S`, or `X` when the system pauses would complete a valid program name - which the Call Paragraph function would act upon and recover from storage through the Merge Interpreter.

But there would be problems if the operator made a mistake in keying, and entered a character not in the list of acceptable responses. If the letter `D` were entered, for example, the system would 'see' the command `ACCTD~G`, which is not the name of a valid program. The system would therefore fail at that point.

Building a default into the command would avoid this happening:

```
ACCT(Pause for input)~G  ACCTX~G
```

In the event that an invalid response were entered, the system would route through to the default program ACCTX, which would be a routine for closing the system own 'tidily' or routing it back for another - and hopefully valid - response.

22.7 The 'GOTO LABEL' facility increases the scope of programs

Bearing many similarities to the CALL MERGE FILE function, GOTO LABEL, is a facility which allows MICROSCRIPT to advance through an applications program file for a 'label'. Having found this point, the system will act upon (execute) the whole program from that point, or until a subsequent 'goto label' instruction is encountered.

Note that GOTO LABEL is an 'internal' command in that it accesses part of the file in which it is embedded. CALL MERGE FILE, in contrast, calls up sections of other files.

The GOTO LABEL function makes it possible to design systems with extensive logic 'routes' through a program; the precise path taken by the system logic depending on the outcome of an earlier stage of the program.

GOTO LABEL could be used, for example, to handle a sub-menu within an application. The response to a screen prompt completes a label and routes the system through to the point required.

In the following example, a small business system is being set up. The data entry routines are controlled through a menu within the program file:

```
1) Home sales
2) Export sales
3) Purchase invoices

Enter option and press [ENTER]
```

A suitable string of commands to handle this task could be:

!	Pause for keyboard entry
Function 46 followed by 1	Store response in abbreviation 1.
[ESC][	Clear any characters on right hand side of response.

`OPTION`	'Core' of label.
[ESC]# n followed by 1	Recall the contents of abbreviation 1.
@	Go to label defined in two previous steps.
`INVALID`	Default label.
@	Go to label INVALID.

These would be displayed as a string of visible equivalents:

```
!~N1~COPTION.~01@INVALID@
```

Note that the labels corresponding to the sections of the program that can be identified through the menu would be:

```
OPTION.1   OPTION.2   OPTION.3   INVALID
```

If any digit other than 1, 2, or 3 is entered when the system pauses - 6, for example -the system tries to route through to a label which does not exist (in this illustration, `OPTION.6`). Being unable to locate this label, Microscript takes the first default label that it finds - `INVALID` - and executes any routines which are held between that label and the next set of delimiters, a pair of closing brackets.

22.8 Numeric values from programs may be stored...

There is every likelihood that a numeric value generated in a calculation will be required elsewhere in the same application. The numeric store facility (discussed in section 16.3 on the calculator) can be used within the Microscript applications programs.

At the point in the program where data has to be stored from a calculator field, enter `S` followed by a number from 0-9 (there are ten identifiable numeric stores).

...and recovered elsewhere in an application

The cursor should be ready to acccept an input in the calculator field, where the contents of a numeric store are to be recovered. The next sequence in the program should be `R`n (where n is the number of the store holding the data).

22.9 Alternate modes can be used to effect in programming

Of the eight alternative modes supported by Microscript, four are of particular relevance to applications programming.

MODE 3 Defeat all visible equivalents

This was discussed in section 22.1

MODE 4 Invisible Mode

This mode of operation was designed specifically for writing applications programs. While it is in force, all write- to-screen operations are suspended. The program can then handle tasks such as line delete and screen clearing in the 'background'. The result is a more professionally presented proram, without rapid cursor movements on the screen during the manipulation of text and data.

MODE 6 Hold format during merge

When text is merged from another document or an abbreviation memory, by default, it will format to the margin settings applicable to the point of the merge.

Mode 6 overides this default, and allows the incoming text to retain whatever format had previously been assigned to it. An indented quotation from a text, for example, would lose its indentations on merging unless the mode had been selected.

MODE 7 Learn mode

This was discussed in Section 22.2

23.0 Preparing disks for use

Computer disks are of two types: hard and floppy disks.

23.1 Hard disks

Hard disks are the most effective method of data storage currently available for use in an office environment: they enable a large system to be implemented in a minimum of space and with very rapid response time. Additionally they do not require any of your attention.

All maintenance must be carried out by technically qualified personel.

The storage capacitites of a hard disk make the removal of the storage medium unnecessary.

23.2 Floppy disks

Floppy disks are widely implemented on microcomputers. Data is stored on a flat disk of recording material which is rotated inside a sleeve. Types of sizes available: 3″, 3.5″, 5.25″ and 8″.

Each disk has a central locating hole and a window into which the recording head is moved during loading.

The AMSTRAD CPC464 uses the 3″ disk.

23.3 Floppy disks require careful handling

If the machine is switched off accidentally before removing a disk, do not leave the disk in the machine indefinitely.

Never allow fingers or other objects near the oblong window on each side of the disk.

Always insert the disk into the drive carefully and close the door gently.

Never remove a disk in the middle of a job.

Keep disks well away from coffee, tea, cigarettes and dust.

Use a felt tip pen when labelling disks to avoid damage to the surface.

Store disks in an upright or semi-upright position in a fire-proof, damp-proof container.

ALWAYS make security copies of important disks and store these separately. Remember that it is not the value of the DISK that is at stake, but the value of all the information that you have stored on it!

24.0 Functions available with microscript

This section of the manual contains a full description of the operational effect of each function.

#1 [ENTER] This function operates in a similar way to the carriage return on a conventional typewriter. It does not affect the text on the screen, so can be used from any position on a line. This function is invalid when MICROSCRIPT is in Calc mode.

#2 DELETE The cursor moves back one space to the character to be deleted and then erases that character.

#3 CURSOR LEFT The cursor moves one place to the left.

#4 CURSOR RIGHT Moves the cursor one place to right.

#5 CURSOR UP Moves the cursor up one line from its current column position.

#6 CURSOR DOWN Moves the cursor down one line from its current cursor position.

#7 [TAB] Moves the cursor to the next tab position.

#8 PUSH TEXT Inserts spaces anywhere along the cursor line.

#9 PULL TEXT Pulls text leftwards to the cursor erasing unwanted spaces.

#10 [TAB] RIGHT Moves the cursor to the first character in the next word to the right.

#11 [TAB] LEFT Moves the cursor to the last character in the previous word.

#12 DELETE LINE Deletes the cursor line causing text to move up.

#13 OPEN NEW LINE Inserts blank lines to allow new text to be added.

#14 GO TO RIGHT END OF LINE Moves the cursor to one space to the right of last character on the cursor line.

#15 GO TO LEFT END OF LINE Moves the cursor to the left hand end of the line.

#16 GO TO TOP OF SCREEN	Moves the cursor to the top of the screen.
#17 GO TO BOTTOM OF SCREEN	Moves the cursor to the bottom of the screen.
#18 CONVERT TO CAPITALS	Converts all characters to the right of the cursor to upper case.
#19 CONVERT TO LOWER CASE	Converts all characters to the left of the cursor to lower case.
#20 CUT & PASTE MARK	Marks the top left hand corner ofthe area to be processed during Cut and Paste.
#21 CUT & PASTE BLANK	Marks the bottom right hand corner of the area to be processed. The text is then copied into the buffer and the original area left blank.
#22 CUT & PASTE MARK & LEAVE	Similar to function ·21 but the area marked is left on the screen once it has been copied into the buffer.
#23 CUT & PASTE MARK & REMOVE	Marks the bottom right hand corner of the area to be processed. But once the ext has been copied, the area is deleted.
#24 CUT & PASTE OVERLAY	Places the text held in the buffer into a document but overwrites existing text.
#25 CUT & PASTE INSERT	Inserts the text held in the buffer into a document.
#26 CUT & PASTE ELBOW	Inserts the text held in the buffer into a document by elbowing existing text, so that it is not overwritten.
#27 CENTRE LINE	Reformats all text centrally under the control of the current ruler.
#28 REFORMAT TEXT	Reformats all text under the control of the current ruler.
#29 STORE RULER	Validates a ruler, then stores it in a ruler memory overwriting the existing ruler.
#30 RECALL RULER	Recalls a ruler from a specific memory.
#31 USE RULER	Instructs MICROSCRIPT to use a ruler contained in a specific memory.

#32 UNDERLINE	Underlines all the text on the cursor line.
#33 UNDERSCORE	Underscores all the text on the cursor line.
#34 PAGE FEED	Feeds the cursor to the top of the screen enabling you to scroll through a document rapidly.
#35 WIPE RIGHT	Erases text on the cursor line to the right of the cursor.
#36 WIPE LEFT	Erases text on the cursor line to the left of the cursor.
#37 DELETE WORD	Deletes part or all of a word.
#38 GO TO TOP LEFT	Sends the cursor to 'HOME' - the top left HAND OF SCREEN.
#39 CALL MERGE FILE	Merges a disk file with the document being edited.
#40 ENTER INSERT MODE	Enters insert mode.
#41 ELBOW TEXT	Moves text to the right of the cursor to the next line.
#42 EXIT INSERT MODE	Leaves insert mode.
#43 GO TO TEXT END	Moves the cursor to the end of the screen.
#44 EXPAND LINE	Right justifies all text from the cursor line to the end of a paragraph.
#45 FIND TEXT	Finds a piece of text in a file.
#46 STORE ABBREVIATION	Stores text in an abbreviation memory.
#47 RECALL ABBREVIATION	Recalls text from an abbreviation memory.
#48 ENTER MODE	Enters an alternate mode
#49 EXIT MODE	Leaves an alternate mode.
#50 QUIT DOCUMENT	Leaves a document without saving it.
#51 RECALL DOCUMENT NAME	Recalls a specific document to the screen.
#52 EXIT DOCUMENT SAVE	Save a document.

#53 GO TO TOP OF DOCUMENT	Return to the top of the document.
#54 CALL SIF FILE TEXT	Calls the SIF file to the screen.
#55 LIFT LINE OF TEXT	Stores a line of text for inclusion elsewhere in document.
#56 PLACE DOWN LINE OF TEXT	Places a line of text, stored by function 55, into a document at the current cursor position.
#57 GOTO BOTTOM OF DOCUMENT	Takes the cursor to the bottom of the current document

25.0 Appendix I

25.1 Standard ASCII Codes

Char	Code	Char	Code	Char	Code	Char	Code
^@	0	Space	32	@	64	\	96
^A	1	!	33	A	65	a	97
^B	2	"	34	B	66	b	98
^C	3	#	35	C	67	c	99
^D	4	$	36	D	68	d	100
^E	5	%	37	E	69	e	101
^F	6	&	38	F	70	f	102
^G	7	'	39	G	71	g	103
^H	8	(	40	H	72	h	104
^I	9	)	41	I	73	i	105
^J	10	*	42	J	74	j	106
^K	11	+	43	K	75	k	107
^L	12	,	44	L	76	l	108
^M	13	-	45	M	77	m	109
^N	14	.	46	N	78	n	110
^O	15	/	47	O	79	o	111
^P	16	Ø	48	P	80	p	112
^Q	17	1	49	Q	81	q	113
^R	18	2	50	R	82	r	114
^S	19	3	51	S	83	s	115
^T	20	4	52	T	84	t	116
^U	21	5	53	U	85	u	117
^V	22	6	54	V	86	v	118
^W	23	7	55	W	87	w	119
^X	24	8	56	X	88	x	120
^Y	25	9	57	Y	89	y	121
^Z	26	:	58	Z	90	z	122
^[	27	;	59	[	91	{	123
^\	28	<	60	\	92	|	124
^]	29	=	61	]	93	}	125
^.	30	>	62	^	94	~	126
^_	31	?	63	_	95	Del	127

25.2 Program function characters

Function		Code	Function		Code
1	Return	!	31	Use ruler 'n'	?
2	Delete	"	32	Underline	@
3	Left	#	33	Underscore	A
4	Right	$	34	Page feed	B
5	Up	%	35	Wipe right	C
6	Down	&	36	Wipe left	D
7	Tab	_	37	Delete word	E
8	Push	(	38	Goto top left	F
9	Pull	)	39	Call merge file	G
10	Tab right	*	40	Enter insert mode	H
11	Tab left	+	41	Elbow line	I
12	Delete line	,	42	Exit insert mode	J
13	Open line	-	43	Goto bottom of text	K
14	Goto right end	.	44	Expand line	L
15	Goto left end	/	45	Find text	M
16	Goto top	0	46	Store abbreviation	N
17	Goto bottom	1	47	Recall abbreviation	O
18	Convert caps	2	48	Alt. mode on	P
19	Convert lower	3	49	Alt. mode off	Q
20	C/P mark	4	50	Quit - no save	R
21	C/P blank	5	51	Recall filename	S
22	C/P leave	6	52	Save document	T
23	C/P remove	7	53	Goto top of doc.	U
24	C/P overlay	8	54	Call SIF text	V
25	C/P insert	9	55	Goto bottom of doc.	W
26	C/P elbow	:	56	Goto prev. screen	X
27	Centre line	;	57	Goto label	Y
28	Reformat	<	58	Stop merge	Z
29	Store ruler 'n'	=	59	Pause for kbd input	[
30	Recall ruler 'n'	>	60	Defeat vis. equiv.	\

26.0 Appendix II - Error Messages

There are three types of error message:

User errors — MICROSCRIPT informs you that you have made an error.

Bugs — Conditions may arise which unearth a fault in the programming. Errors of this kind will be reported in the form:

```
Error 64/960
```

This type of message is usually followed by an 'English Language' one line explanation. When such an error occurs you will automatically be taken out of Microscript back to the operating system. You should note the following:

The date of issue
The exact text of the error message
The operation being performed at the time

The report these facts by phone to your supplier.

External errors — Errors may occur due to insufficient space on the disk or in the disk directory. These errors will be reported by the operating system and having remedied the cause of the error, Microscript can be run again.

INDEX

MICROSCRIPT functions - *Create and Edit*

Function	Operation	Key Sequence
1	‹Enter›	**[ENTER]**
2	Delete	**[DEL]**
3	Cursor left	←
4	Cursor right	→
5	Cursor up	↑
6	Cursor down	↓
7	‹Tab›	**[TAB]**
8	Push text	**[CTRL] N**
9	Pull text	**[CLR]**
10	Tab right	**[CTRL] T**
11	Tab left	**[CTRL] R**
12	Delete line	**[CTRL] Y**
13	Insert lines	**[ESC] O**
14	Go to right end of line	**[CTRL]** →
15	Go to left end of line	**[CTRL]** ←
16	Go to top of screen	**[ESC]** ↑
17	Go to bottom of screen	**[ESC]** ↓
18	Convert to Capitals	**[ESC] >**
19	Convert to lowercase	**[ESC] <**
27	Centre line	**[ESC] C**
29	Store ruler 'n'	**[ESC] S‹n›**
30	Recall ruler 'n'	**[ESC] R‹n›**
31	Use ruler 'n'	**[ESC] U‹n›**
32	Underline with '-'	**[ESC] -**
33	Underscore with '_'	**[ESC]** _
34	Page feed	**[ESC] P**
35	Wipe right	**[ESC] [**
36	Wipe left	**[ESC]]**
37	Delete word	**[CTRL] V**
38	Go to top left of screen	**[CTRL]** ↑
39	Call merge file	**[ESC] G**
40	Enter insert mode	**[ESC] I**
41	Elbow text to next line	**[ESC] [RETURN]**
42	Exit insert mode	**[ESC] [ESC] I**
43	Go to text end	**[CTRL]** ↓
45	Find text	**[ESC] F**
46	Store abbreviation 'n'	**[ESC] #** ‹n›
47	Recall abbreviation 'n'	**[ESC] @** ‹n›
48	Enter mode 'n'	**[ESC] M‹n›**
49	Exit mode 'n'	**[ESC] [ESC] M‹n›**
50	Quit document	**[ESC] Q**
51	Recall document name	**[ESC] ?**
52	Save document	**[ESC] E**
53	Goto top of document	**[ESC] T**
55	Lift a line of text	**[COPY]**
56	Place down a line of text	**[ESC] [COPY]**
57	Go to bottom of document	**[ESC] B**

MICROSCRIPT functions -*Reformat*

Function	Operation	Key Sequence
1	‹Enter›	**[ENTER]**
3	Cursor left	←
4	Cursor right	→
5	Cursor up	↑
6	Cursor down	↓
7	‹Tab›	**[TAB]**
12	Delete line	**[CTRL] Y**
13	Insert line	**[ESC] O**
14	Go to right end of line	**[CTRL]** →
15	Go to left end of line	**[CTRL]** ←
16	Go to top of screen	**[ESC]** ↑
17	Go to bottom of screen	**[ESC]** ↓
20	Cut & paste mark	**[ESC] (**
21	Cut & paste mark blank	**[ESC]) B**
22	Cut & paste mark leave	**[ESC]) L**
23	Cut & paste mark remove	**[ESC]) R**
24	Cut & paste overlay	**[ESC] * O**
25	Cut & paste insert	**[ESC] * I**
26	Cut & paste elbow	**[ESC] * E**
28	Reformat	**[ESC] J**
29	Store ruler 'n'	**[ESC] S**‹n›
30	Recall ruler 'n'	**[ESC] R**‹n›
31	Use ruler 'n'	**[ESC] U**‹n›
32	Underline with '-'	**[ESC] -**
33	Underscore with '_'	**[ESC]** _
34	Page feed	**[ESC] P**
35	Wipe right	**[ESC] [**
36	Wipe left	**[ESC]]**
38	Go to top left	**[CTRL]** ↑
39	Call merge file	**[ESC] G**
43	Goto end of text on screen	**[CTRL]** ↓
44	Expand line	**[ESC] X**
45	Find text	**[ESC] F**
50	Quit document, no save	**[ESC] Q**
51	Recall document name	**[ESC] ?**
52	Exit document, save	**[ESC] E**
53	Go to top of document	**[ESC] T**