BRITISH BROADCASTING CORPORATION

MICROCOMPUTER SYSTEM

User Guide

PART 2

BBC Microcomputer System User Guide Part 2

Part no 423001
Issue no 1
Date October 1983

WARNING

This computer is a Class 1 device, and therefore requires connection to a protective earth. In order to ensure this the mains power cable plug shall only be inserted into a socket outlet provided with a protective earth contact. DO NOT defeat the earth grounding protection by using an extension cable, or autotransformer, without a protective ground conductor. Failure to ground the computer properly can result in serious personal injury.

The product described in this manual and products for use with it are subject to continuous development and improvement. All information of a technical nature and particulars of the product and its use (including the information and particulars in this manual) are given by Acorn Computers in good faith. However, it is acknowledged that there may be errors or omissions in this manual. A list of details of any amendments or revisions to this manual can be obtained upon request from Acorn Computers. Acorn Computers welcome comments and suggestions relating to the product and this manual.

All correspondence should be addressed to:

Acorn Computers Corp
400 Unicorn Park Drive
Woburn, Massachusetts, USA

Within this publication the term 'BBC' is used as an abbreviation for 'British Broadcasting Corporation'.

First published 1983
Published by Acorn Computers Limited, Cambridge, UK
Typeset by Bateman Typesetters, Cambridge, UK
Printed by Saunders & Williams (Printers) Limited, Croydon, Surrey, UK

LIMITED WARRANTY

Acorn Computers Corp warrants, to the original consumer purchaser, that this British Broadcasting Corporation Microcomputer shall be free of defective materials or workmanship for a period of 90 days from the date of purchase from Acorn or an authorized BBC Microcomputer dealer. If, during the 90 day warranty period, this product should contain defective materials or workmanship, Acorn will, at its option, repair or replace this product at no additional charge except as set forth below. This limited warranty does not include services to repair damage to the product resulting from: accident, disaster, misuse, abuse, non-Acorn modification of the product, defects or damage caused by unauthorized personnel or defects in non-authorized replacement parts.

If you believe that your computer is malfunctioning because of a defect of materials or workmanship which is covered by this warranty, service under this limited warranty can be obtained by taking the product to an authorized Acorn Service Center. If you believe that you are entitled to service under this limited warranty, you must bring with you proof of purchase date. Return of your warranty registration card, while helpful to us in servicing our customers, is not a condition precedent to limited warranty coverage. Write to Acorn Computers Corp, 400 Unicorn Park Drive, Woburn, Mass or call (617) 935-1190 to obtain the name of the closest authorized Acorn Service Center or for further information.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO A PERIOD OF 90 DAYS FROM THE DATE OF PURCHASE AND NO WARRANTIES, WHETHER EXPRESS OR IMPLIED, WILL APPLY AFTER THIS PERIOD. Some states do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you.

Acorn's sole obligation with respect to any failure of or defect in the computer is limited to its obligations to repair or replace damaged equipment under this limited warranty. IN NO EVENT WILL ACORN COMPUTERS CORP BE LIABLE FOR LOSS OF USE OF THE COMPUTER OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES OR DAMAGE INCURRED BY THE CONSUMER OR ANY OTHER USER EVEN IF ACORN OR ITS AGENTS HAVE BEEN ADVISED OF A POSSIBILITY OF SUCH DAMAGES. Some states do not allow the exclusion or limitation of incidental or consequential damages so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

FEDERAL COMMUNICATIONS COMMISSION

RADIO FREQUENCY INTERFERENCE STATEMENT

Warning: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers etc) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. In order to comply with the limits for a Class B computing device, all external data cables currently supplied by Acorn Computers Inc and which are connected to the computer are shielded; all ribbon cables are shielded, and individual circular data cables are shielded and incorporate a ferrite sleeve adjacent to the connector which connects to the computer. It is the user's responsibility to ensure that cables other than those supplied by Acorn Computers Inc incorporate these design features, and failure to do so may result in harmful interference to radio and television reception. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Re-orient the receiving antenna.

Relocate the computer with respect to the receiver.

Move the computer away from the receiver.

Plug the computer into a different outlet so that the computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:

How to Identify and Resolve Radio-TV Interference Problems

This booklet is available from the US Government Printing Office, Washington, DC 20402, stock no 004-000-00345-4.

Contents

Disk Filing System User Guide

Introduction 1

About this manual 1

1 What is a disk system? 2

A disk drive 2
What the disk drive does 3
Disk Filing System 4
Controlling the filing system 5
Summary 6

2 Getting going 7

3 Disks 1Ø

Handling 1Ø
Preventing accidental erasure 12
Copying information 12
Proceed as follows 13
Tracks, sectors and bytes 17
What is formatting? 17

4 Disk files 19

File specifications 19
Drive numbers 2Ø
Directories 2Ø
File names 21
Multi-file operations 21
Auto-start facilities 22
Library directory 23

5 The BREAK key 24

6 The filing system commands **25**

Command 25
Purpose 25
Example 25
Description 25
Associated commands 26
Demonstration program 26
Notes 26
Diagrams 26

7 The filing system utilities **63**

8 Random access files **68**

9 Using the filing system in assembler **73**

General principles 73
OSFIND 74
OSARGS 74
OSFILE 75
Read/write one byte 77
Read/write a group of bytes 77
Read/write a sector 8Ø

1Ø Changing filing systems **81**

11 Error messages **82**

12 Technical information **85**

18-bit addressing 85
Disk catalog 85
File system initialize and !BOOT 86
Link facilities 87

13 Filing system command summary **89**

Speech System User Guide

Introduction to the Speech System 95

Access to the Speech System 95
In BASIC 95
In Assembly Language 95
Using the Speech System 96

1 Using the Speech System from BASIC 97

2 The SOUND command in detail 99

Using absolute addresses 100

3 Making your own words 101

Introduction 101
Program for sending speech data to the Speech System 102
Loading speech data for the word 'zero' into memory 103

4 Using the Speech System in Assembly Language 105

Introduction 105
OSWORD call with A=&07 105
OSBYTE calls &9E and &9F 107
OSBYTE call with A=&9E – read from the Speech Processor 107
OSBYTE call with A=&9F – write to the Speech Processor 107

5 Technical information 109

Introduction 109
The Speech Processor 109
The Speech Processor registers 109
Command register 110
FIFO buffer 110
Data register 110
Status register 110
The Speech Processor commands 111
Read byte 112
Read and branch 112
Load address 112

Speak 112
Speak external 112
Reset 113
Speech data ROMs 113
Speech data 114
Word PHROM VM61ØØ2 116
VM61ØØ2 Word PHROM format 116
Word pointers 117
Word data 117

Appendix A **118**

Full list of words in Word PHROM VM61ØØ2 (alphabetic)

Appendix B **124**

Full list of words in Word PHROM VM61ØØ2 (by word number)

Appendix C **13Ø**

Speech parameter data coding

Appendix D **132**

References

Into VIEW

1 What is word processing? **137**

2 The VIEW word processor **139**

BASIC 139

3 Word processing with VIEW **14Ø**

Switching on 14Ø
The command mode display 14Ø
Text mode 141
The text mode display 141
Screen and page 141
Typing text 142

4 Screen modes and ruler **146**

The ruler 147
Do-it-yourself rulers 148
Further experiments with rulers 149

5 Tabbing **15Ø**

Tabbing after typing 15Ø
TAB characters 151
Text outside the ruler 152
Tables and formatting 153

6 Saving and loading files **154**

Disk systems 154
Setting up 154
Formatting 154
Using disks 154
Locking files 156
Help! 156
Cassette tape 156
Recording files 157
Reading files 157
Printing 157
Names of files 158

7 Printing **159**

General procedure for printing 159
PRINT 161
SHEETS 161
Editing procedures 162
SCREEN 162
Highlights 163
Resetting highlight codes 163
Printing from cassette 164

8 Moving and changing text **165**

9 Stored commands and page layout **168**

Book and report work 169
Number registers 172

10 Macros **173**

Modified macros 174
Macros for mail shots 178
Automatic layout 179

11 Change . . . replace . . . search **183**

REPLACE 184
SEARCH 185
Limited searching and changing 185
Finding and changing phrases 185

12 Special facilities **187**

COUNT 187
FORMAT 187
Editing BASIC programs 188

13 Continuous processing **190**

The EDIT method 190
Finishing 192

VIEW Guide

Introduction **197**

Commands 197
Presentation of commands 197
Useful hints 198

1 Start-up **199**

Command mode 199
Text mode 200

2 Immediate commands **201**

Moving the cursor 201
Lines 202
Deletion 203
Insertion and swapping case 204

Markers 205
Block operations 206
The ruler 207
Formatting and justification 209
Formatting and margins 210
TAB stops and characters 211

3 Stored commands 213

Lines and page length 214
Ejecting pages 215
Page layout 216
Headers and footers 220
Macros 222
Macros with parameters 223
Highlight 1 and 2 227
Number registers 229

4 Command mode commands 230

Screen modes 230
Clearing text from memory 230
Using the disk 231
Editing BASIC programs 232
Continuous processing (disks only) 233
Using cassette tape 235
Word counting and formatting 236
Finding words in the document 237
Printing 240

Disk Filing System User Guide

Introduction

Before you start using the Disk Filing System, check that you have all the following items:

- A single disk drive, or a dual disk drive (two single drives in the same box). Both single and dual drives should be fitted with ribbon cable and power cable.
- An introductory and utilities disk.
- A BBC Microcomputer fitted with a disk interface, which should only be fitted by an authorized dealer.

If any of the necessary items are missing then contact your supplier, quoting the order number given to you when you placed your order. The number also appears on the dispatch label.

About this manual

You will notice that some letters, words and phrases in this manual have been printed differently from the rest of the text. This is to help you to tell the difference between explanatory text, words which appear on the computer screen (including BASIC keywords), and certain keys on the computer keyboard.

- Ordinary text appears like this, or *like this* for emphasis.
- Text typed in at the computer or displayed on the screen or BASIC keywords appear `like this`.
- Words like **RETURN** mean that you press the key marked RETURN rather than actually type the letters R E T U R N.

1 What is a disk system?

If you have never used a computer with disks before there are one or two new concepts which you need to learn.

A disk drive

As you probably know, computers have internal memory called Random Access Memory or RAM. When you type in your program it is stored in RAM. However, when you switch off the computer, everything stored in RAM is lost, so if you need the program again, you have to retype it. To overcome this problem the computer must be able to transfer the contents of RAM into some form of permanent or 'non-volatile' storage before you switch it off. The User Guide which comes with your BBC Microcomputer describes how to use a cassette recorder for this purpose. Transferring a program from RAM to tape is called *saving* it, transferring from tape back to RAM is called *loading* it. The disadvantages of using tape are:

- The process of saving and loading is quite slow.
- You need to keep track of where on the tape each piece of information is, so that you do not record over it.
- You have to wind the tape to the right place yourself.
- Winding from one end of the tape to the other is slow.
- It is not possible to wind the tape to a particular point accurately.

A disk system does not have these disadvantages. To help you to understand how a disk system works, we shall draw some comparisons with a filing cabinet. A disk system always includes at least one disk drive. The BBC Microcomputer's disk drives are buff-colored metal boxes approximately 15.5cm × 9cm × 24cm. The front of a disk drive is black. On the front is the BBC owl, a small red light and a spring flap called the disk drive door. The door can be opened or closed, but must be closed while the disk drive is supposed to be working. These are floppy disk drives, as distinct from fixed disk drives, which you may have heard of. The disk drive can be compared to an empty filing cabinet with no drawers in it yet. The dual disk drive box may contain two disk drives, one on top of the other. This text will only refer to one drive for general descriptions of the Disk Filing System, unless a distinction needs to be made between two drives.

Just as a filing cabinet is pretty useless without drawers, so a disk drive cannot do much without disks. The disks used with the BBC Microcomputer are 5¼" soft-sectored floppy disks. They can be inserted and removed from the disk drive via the slot in the front. There is one right way of inserting them, and several wrong ways. As in the case of the filing cabinet drawers, putting them in the wrong way is a waste of time. Fig 1 shows the correct way to insert a disk. To stress this:

Disks should be inserted with the label upwards and with the edge nearest the label in your hand. The edge opposite to the label and the read/write slot of the disk go in first.

Fig 2 shows and names the various parts of a disk and chapter 3 gives more detailed information about them. The disks hold the information. Just as you can put different cassettes in a cassette recorder, you can also put different disks into a disk drive; but the computer can only read information from one disk at a time. A disk may have lots of different groups of information on it, and these groups are called 'files'. Files can have any information in them. Typical examples would be one of your programs or some data generated by a program which you wish to keep.

Returning again to the comparison with a filing cabinet, opening a drawer and throwing all the papers into it at random would make it difficult to find them again. To solve this problem, people usually put dividers into a filing cabinet drawer, and often these are labelled alphabetically. The result is that the information is grouped so that you can find it again quickly. The same principle is followed when the computer puts information on to a disk. When you first buy disks, they are blank – like empty drawers. Before the computer can put any information on them, the disks must first be prepared by having marks put on them, which divide the disks into sectors. A 'sector' is the name given to a set of equal divisions created on the disk by the computer. (See Fig 5.) This operation is called 'formatting' and is fully described in chapter 7.

What the disk drive does

When you insert the disk into the disk drive and close the drive door, a rotating boss engages with the central hole in the disk and spins the magnetic disk inside its protective jacket. (Do not confuse the protective jacket with the disk envelope – see Fig 2.) In order to read or write information on to the disk the disk drive has a 'read/write head'. This

head is designed to move in and out along the 'head slot' in the disk jacket. This head actually rests on the surface of the magnetic disk as it rotates inside the jacket. When you want to read some of the information on the disk, you give the computer the name of the file containing that information. The computer will move the read/write head to the sector on the disk where the start of the information in the named file is recorded. This is equivalent to you opening the filing cabinet drawer, looking along the dividers until you find the one you want and then preparing to remove the relevant file for reading.

At this point it is worth noting that your files may be too large to fit into the fixed size of one sector. This is no problem. A file always begins in a new sector but may occupy a number of sectors following the first. Each sector can hold up to 256 characters or 'bytes'.

Disk Filing System

The main disadvantage of using a cassette recorder to store information is that you have to control the cassette recorder and keep track of the information on it.

When using a disk this is all done for you by the 'Disk Filing System'. The Disk Filing System is a machine-code program produced by the computer manufacturer. On the BBC Microcomputer it is stored in a special kind of memory inside the computer called 'Read Only Memory' or ROM. The program is not lost when you switch the computer off. Once installed, it is always there. All the actions of the disk drive are controlled by the computer using this program. When you prepare new disks by formatting them this is done by the Disk Filing System. When you **SAVE** one of your BASIC programs the Disk Filing System does the following:

- Starts the disk drive working.
- Finds a free place on the disk big enough for your program.
- Makes a note of where it put your program so as to be able to find it again.
- Moves the disk drive's read/write head accurately to the start of the first sector in the free space.
- Transfers a copy of your program from the RAM to the disk.
- Stops the disk drive.

All this is done without you having to think about it and is quite a bit quicker than saving a program on to a cassette tape.

When you save a program you have to give it a name. This is true for the disk system as well as the cassette system. However, the Disk Filing System puts the name to special use. The first two sectors on every disk are reserved for a 'catalog' when the disk is formatted.

The name of your program, referred to as a file name, is written into the catalog together with the number of the sector on the disk where the information starts. (Note it may continue over several sectors.) When you want the file containing your program back again you simply type `LOAD "filename"`. The filing system checks the catalog to find out where on the disk to find the file, and then moves the read/write head to that exact place on the disk. The file is then loaded into the computer's memory (RAM) automatically. This illustrates another advantage of a disk drive. The read/write head can be quickly moved to any point on the disk with great accuracy. (Incidentally, the precision engineering needed to accomplish this explains why disk drives cost so much more than cassette recorders.) Because of this accuracy, a number of other facilities are available besides loading and saving programs. These include the ability to copy, delete, build and rename files. Additional facilities let you examine a disk catalog, restrict access to files or move directly to specific points within a file.

As a final comparison, imagine an automatic filing cabinet where to find something all you have to do is specify the name of a document and paragraph number within it. The filing cabinet drawer opens, the correct divider is selected, the document is located and then presented to you open at the appropriate page. You may see why microcomputers have become so popular in offices.

Controlling the filing system

The filing system controls the disk drive, and we must be able to give instructions to the filing system. Two ways are provided. One is by typing a '*' character, followed by a special command. These are all listed in chapter 6 together with details of their functions. Any of these direct commands can be incorporated in a program if required. Chapter 8 describes the use of a number of BASIC keywords, with special reference to files created on a disk. All these keywords are introduced in the BBC Microcomputer User Guide.

Summary

A disk system includes either one or two disk drives, some disks, a connection to the computer and a machine-code program permanently in the computer called a 'Disk Filing System'.

A disk is inserted into a disk drive where it spins round inside its protective jacket.

The disk drive's read/write head moves in and out along a radius of the disk as it spins around.

The Disk Filing System controls the disk drive and the movement of the head.

Disks are divided into sectors by the filing system and the first two sectors are reserved for a catalog.

Programs and other information are stored on the disk and are given a 'file name'.

By referring to the catalog, the filing system can find any information on a disk associated with a specified file name.

Files can occupy more than one sector.

Procedures are provided to locate particular points in files.

Instructions may be given to the filing system by direct command or from within a program.

2 Getting going

With the power turned off, connect the two cables from the disk drive to the underside of the computer as shown in Fig 1. The plugs are designed so that they will only fit one way. The plug on the power cable is shaped like a rectangle with two adjacent corners cut off. *Do not* force it in the wrong way round. Sometimes the plug on the ribbon cable has a lump on one side which locates into a notch in the socket on the computer. Alternatively, you may have to try both possible ways before you get it right. When the drive is connected, turn on the power and press **BREAK**. The following message, or one very similar, should appear on the screen:

```
Acorn OS

Acorn DFS

BASIC

>
```

This indicates that the Disk Filing System is installed and working OK. Now press the **SHIFT** key and tap the **BREAK** key, then release the **SHIFT** key. The disk drive motor will start turning for a couple of seconds and the red light on the front of the drive will come on. This shows that the disk drive is properly connected and working. If nothing happens, check the connections between the disk drive and computer.

The foregoing assumes that the auto-start option has been set to work when **SHIFT** is held down with **BREAK**. (See chapter 4.)

Insert the introductory and utilities disk into drive Ø. (With dual drives, drive Ø is the one at the top. See Fig 1 for drive numbering.)

Press **SHIFT** and **BREAK** again. This time the following message will appear on the screen:

```
Hello, this is the BBC microcomputer.
This message was automatically loaded.
```

The introductory and utilities disk contains two useful programs:

*FORM80 formats an 80 track disk
*VERIFY verifies a disk

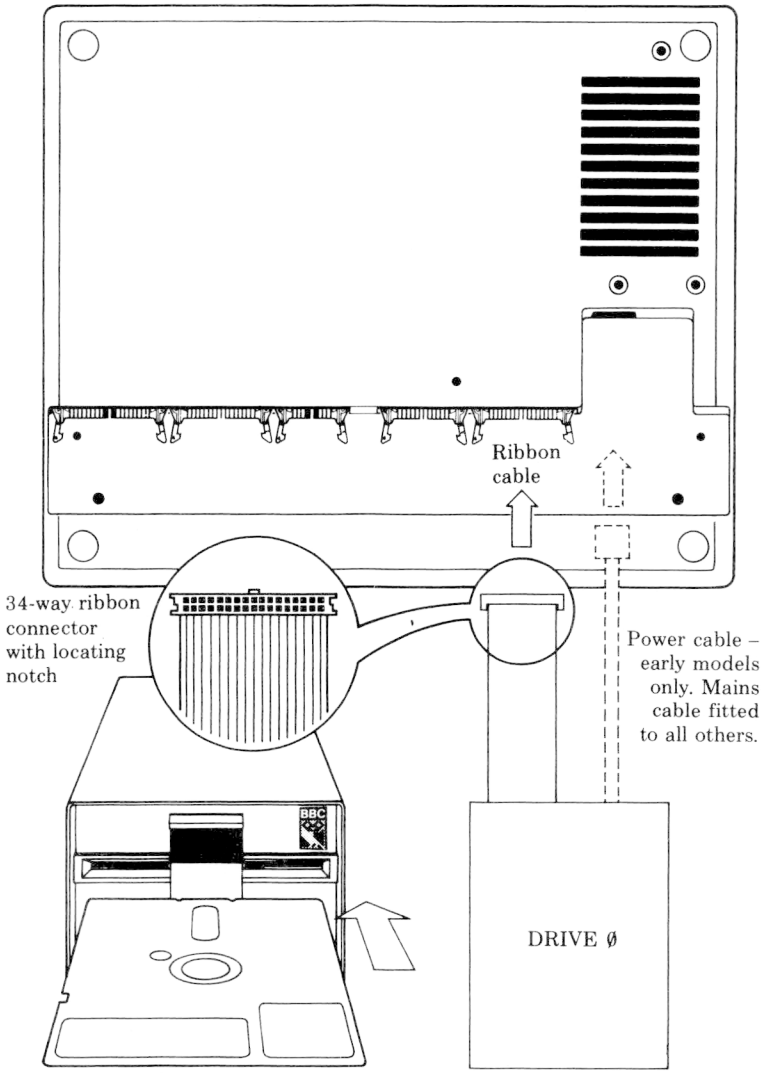
These programs are used by typing their name.

There are also a number of demonstration programs on this disk.

Press the space bar to see them run.

This is an example of a !BOOT file which will be explained in more detail later. Now press the space bar to continue. This will run the series of demonstration programs supplied on the disk. When the > prompt reappears on the screen you can start entering your programs or filing system commands, but before you do that, read the next two chapters, 'Disks' and 'Disk files'.

Fig 1 Connecting the disk drive



3 Disks

The BBC Microcomputer uses 5¼" soft-sectored floppy disks for storing information. You may have heard them referred to as 'floppy disks', 'diskettes', or 'mini-disks'; we will always refer to them simply as disks.

Two types of disk drive are available for the BBC Microcomputer, a single unit (1ØØKbytes) and a dual unit (8ØØKbytes). A disk from a single drive will store about 1ØØ,ØØØ characters, equivalent to 5Ø pages of this book. Disks from a dual drive unit will store about 4ØØ,ØØØ characters each.

Handling

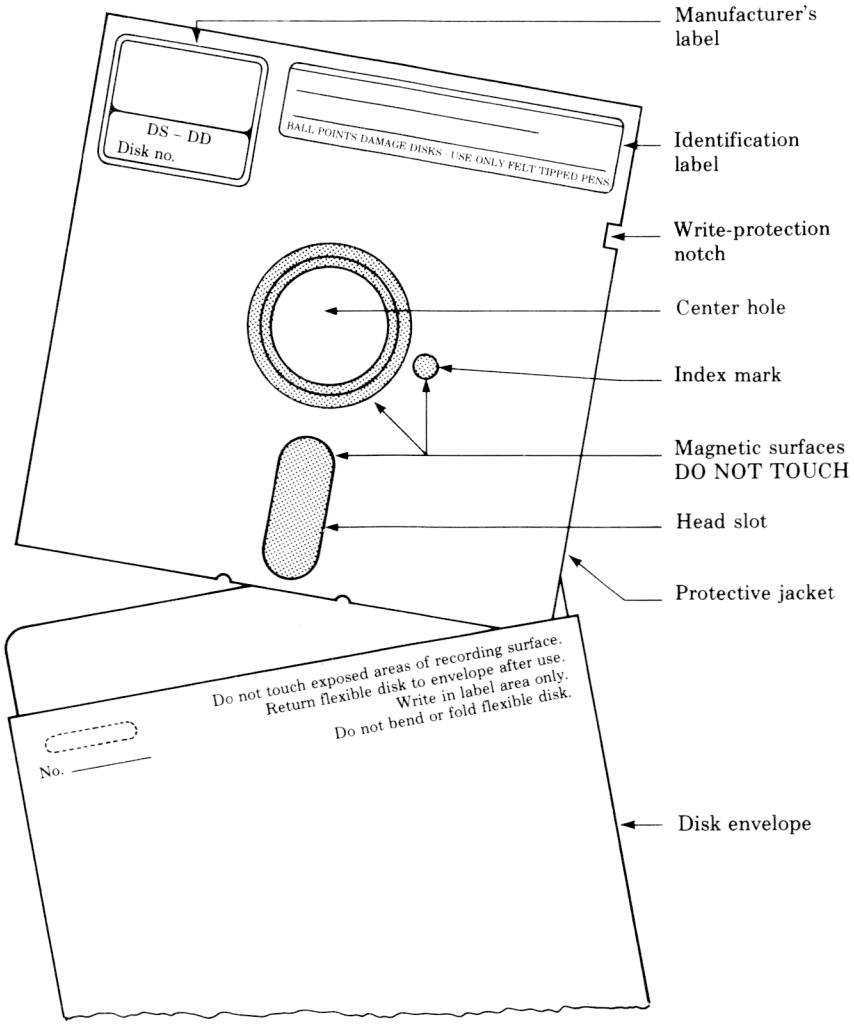
Disks should be handled with care to avoid physical damage or damage to the recorded information. Fig 2 will help you to identify the various parts of the disk that we are referring to. The following guidelines should be observed:

- Do not try to remove the circular magnetic disk from the square, black protective jacket covering it.
- Do not touch the exposed recording surfaces.
- Avoid dust. Put the disks back into their envelopes when they are not in the disk drive.
- Do not bend them, drop them on the floor or put heavy objects on them.
- Keep them in a storage box designed for the purpose.
- Keep them away from strong magnetic fields such as those generated by televisions, monitors, tape recorders, telephones, transformers, calculators, etc.
- Avoid excessive heat, moisture and direct sunlight.
- Only use felt-tipped pens to write on the labels and don't press hard.
- Insert disks into the drive carefully. If it rotates noisily open the drive door and adjust it. If you get an error message such as `Disk fault` or `Drive fault`, then the disk probably hasn't centered in the drive. To overcome this, make sure that the drive spindles are rotating by addressing the drive before closing the drive door. This is especially important when a new disk is inserted.

Information is packed quite densely onto the disk, so it is sensitive to even very small scratches and particles of food, dust, tobacco, etc.

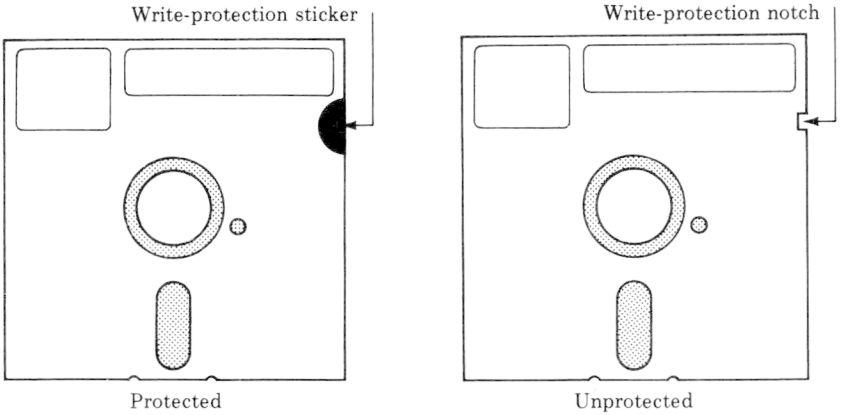
The foregoing is deliberately comprehensive but do not let it frighten you from using the disks. Handled sensibly, a disk will give good service.

Fig 2 A 5¼″ disk



Preventing accidental erasure

You will notice from Fig 2 that there is a small notch in the side of a disk called the 'write-protection' notch. It is used to protect the information on a disk from being overwritten. Every box of disks is supplied with a number of adhesive tabs which can be used to cover the write-protection notch in the disk. The diagram below shows that covering the notch with one of the adhesive tabs will prevent the disk drive from writing to the disk and from deleting anything on it. Reading the existing information on the disk is still allowed.



Copying information

Another way of protecting important information is to keep several copies of it on different disks. Where computers are used in business, industry or other activities which use large volumes of information, a standard routine for this has been evolved. It is often called the 'grandfather, father, son' principle of copying information, and is good practice for anyone using a disk storage system. It works as follows:

Day 1, MASTER copied to GRANDFATHER
Day 2, MASTER copied to FATHER
Day 3, MASTER copied to SON

As you can see, it involves keeping three separate disks, each with a copy of the information from the master disk on it. On day 4 the master would be copied to the grandfather again and so the cycle continues. In business, where information on disk changes from day to day, this regular

routine is important. For personal computing it may not be so vital, but you will want to keep several copies of important programs and information which you have worked hard to produce. The filing system provides two facilities which make copying information easy. These are *BACKUP and *COPY which allow you to copy a complete disk, or specified sections of it. See Figs 3 and 4 and the appropriate sections of chapter 6 for full details.

We suggest that you make a copy of your introductory and utilities disk now!

Proceed as follows

If it has not been done already write-protect the utilities disk with a tab, as described. This is very important otherwise you might lose all the information on the disk.

Insert the utilities disk into drive Ø and type

```
*FORM 80 RETURN
```

Remove the utilities disk and put a new disk in its place. Do not forget to change the disk over. If you format the utilities disk you will lose everything on it.

In reply to the question displayed, type

```
Y
```

What the computer now does is called 'formatting' which prepares the new disk for use with the BBC Microcomputer. A series of numbers are displayed, and when the formatting is finished the word Formatted appears. Now remove the new disk and put the utilities disk back again.

If you have a dual drive, put the new disk into drive 1 (the other drive) and type

```
*ENABLE RETURN
*BACKUP Ø 1 RETURN
```

The computer will now make a complete copy of the introductory and utilities disk on the new disk.

If you have a single drive, type

```
*ENABLE RETURN
*BACKUP 0 0 RETURN
```

then follow the instructions on the screen to insert the utilities (master) disk and the new disk alternately. (See Fig 4.)

If the message

```
Disk read only
```

appears, it means you had the wrong disk in the drive. Start again from *ENABLE.

Fig 3 Copying

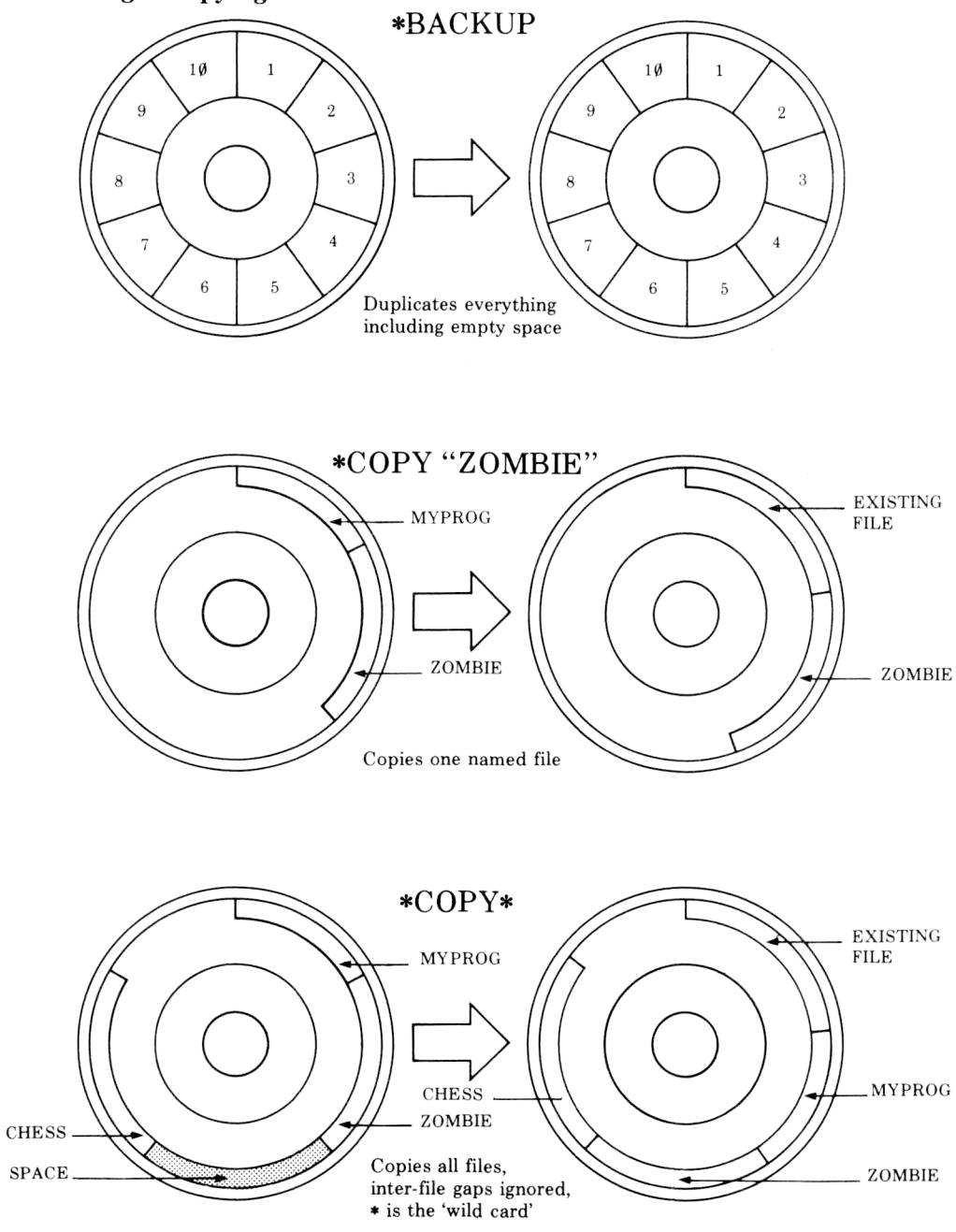
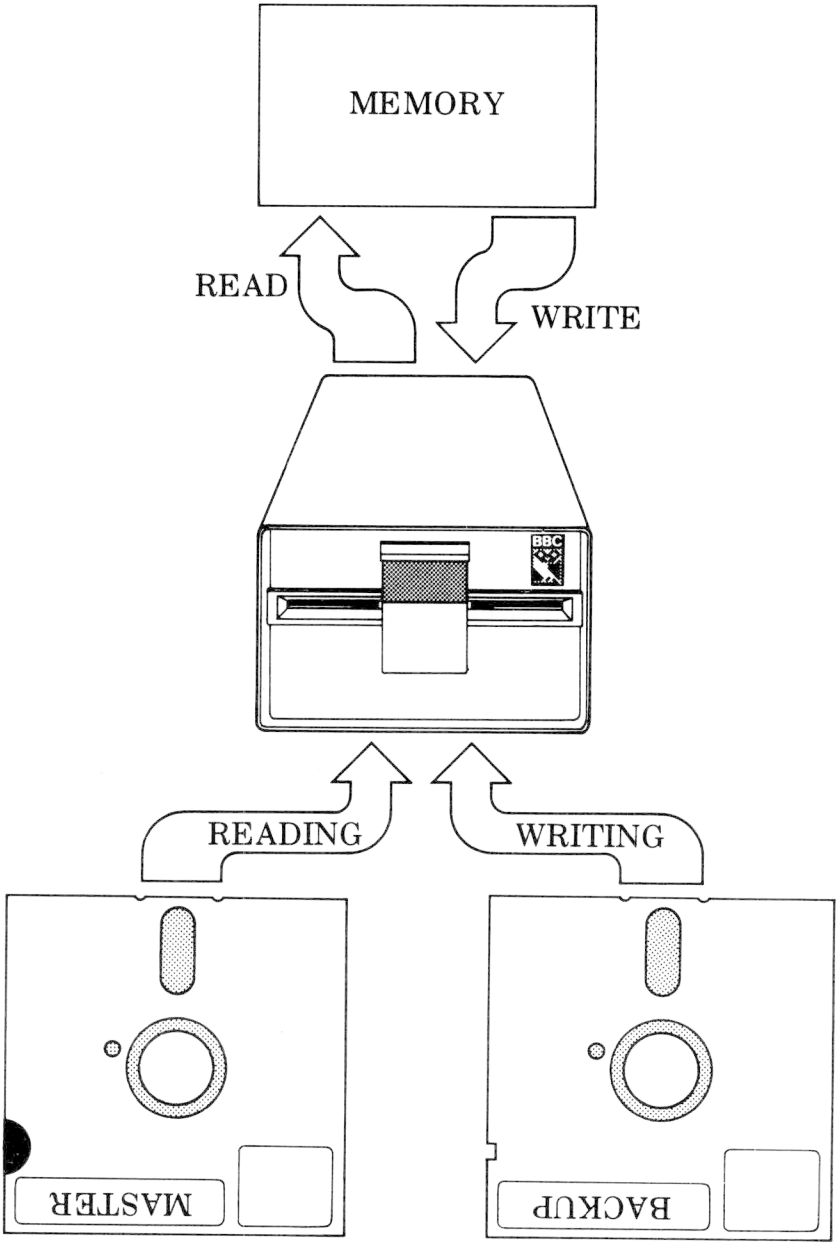


Fig 4 Single disk drive copying



Tracks, sectors and bytes

Information is written on to the disk in concentric circles, called tracks. Each track is divided into ten sectors. Each sector is further divided into 256 bytes. Space on the disk and in the computer's memory is measured in bytes, and one byte corresponds to one character. 256 of the bytes in each sector are available for storing your programs and data. The dual drive unit uses double-sided, double track density disks. Therefore with 8Ø tracks on each side, one disk will hold

16Ø tracks × 1Ø sectors × 256 bytes = 4Ø96ØØ bytes

(or characters) of information.

What is formatting?

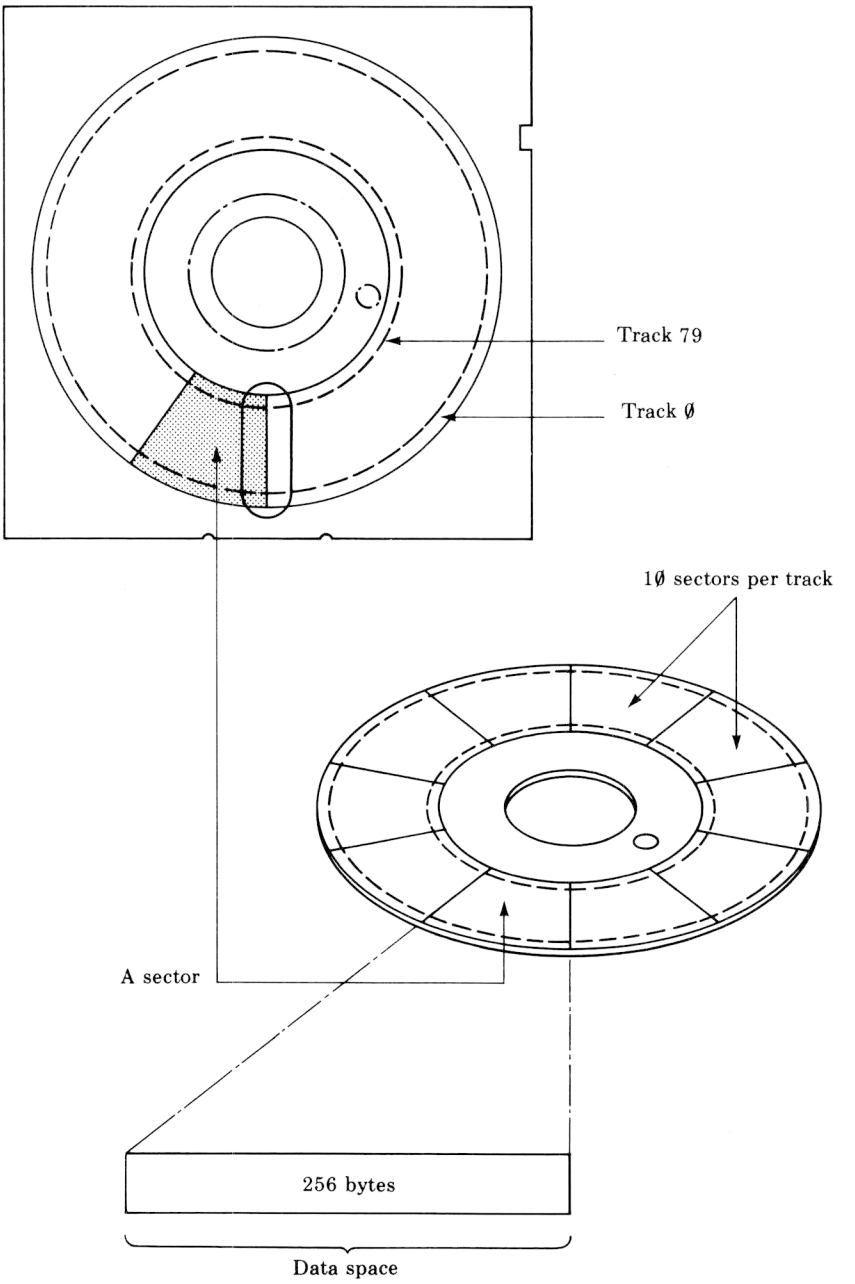
The Disk Filing System automatically records the location of your programs and data on the disk. The first two sectors on track zero of a disk are reserved for this purpose. The 'catalog' of a disk is recorded in these sectors. Whenever you wish to access a piece of information on the disk, the filing system reads the catalog first to find out where on the disk it can be found. The tracks and sectors are the reference marks on disk which make this possible.

Clearly before a disk can be used by the filing system for storing your information it must have these 'reference marks' put onto it. All new disks must be prepared in this way. The process is called 'formatting'. It includes setting up the track and sector format on the disk and creating the catalog. (Full details are provided in the technical information in chapter 12.)

The introductory and utilities disk has a program on it called `*FORM 8Ø` which performs this task for 8Ø track disks. Each sector on each track is given a unique three-digit identifier in the catalog. The first two digits are the track number, the last one is the sector number. The sectors are numbered Ø to 9 and the tracks ØØ to 79.

The important thing to remember is that new disks must be formatted before you can use them with your computer.

Fig 5 Tracks, sectors and bytes



4 Disk files

Probably the first thing you will want to do with the filing system is to record one of your programs onto a disk. You can do this simply by using the SAVE command in BASIC, and the filing system takes care of the rest. (*Note:* Not to be confused with *SAVE described later in this manual.) When you have typed the program into the computer the SAVE command causes it to be copied onto the disk. When SAVEd, the program must be given a name. This is called the file name and is used to identify the program when you want to copy it back from the disk into the computer's memory. Each program SAVEd onto the same drive must be given a unique name. The format of the SAVE command is

```
SAVE  "filename"
```

where "filename" can be up to seven characters. Letters and digits are allowed. The characters

```
#     *     .     :
```

have special meanings which are explained later.

The file name is written into the catalog together with the sector number where the information starts. Next time you refer to the file name, the filing system checks the catalog to see where the information has been placed on the disk, and the old file is deleted and replaced by the new one. The filing system ensures that each new file begins with a new sector.

File specifications

The full specification for a file is

```
:Drive number.Directory.File name
:<drv>.<dir>.<filename>
```

eg:

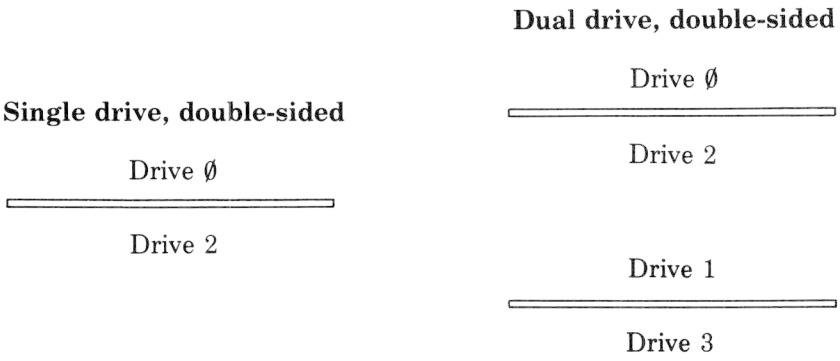
```
:1.Z.MYPROG1
```

Notice the drive number, directory and file name are separated by periods. These are needed so that the computer can distinguish the separate parts of the file specification.

Drive numbers

Drive numbers must be in the range Ø to 3 and preceded by a : (colon). The colon tells the computer: 'This is the start of a file specification, the drive number follows.'

The drives are numbered as shown below. Notice that each side of a double-sided disk is given a separate drive number.



The effect of including the drive number in the full specification is that

`:1.$.MYPROG 1` is a different file from `:2.$.MYPROG 1`

Although the file names are the same, they are on different drives.

Directories

The directory is a single character used to divide the catalog into independent sections. Files of the same name can be created on the same disk with different directories. Although on the same drive,

`:1.$.MYPROG` is a different file from `:1.A.MYPROG`

because the directory is different.

File names

The file name can be up to seven of most of the characters on the keyboard in any combination, except # * : . as we mentioned earlier. When we need to refer to the complete file specification in future we will use the abbreviation <fsp>.

When the filing system is started by pressing **BREAK** or **SHIFT** **BREAK**, the current directory and drive number is always set to drive Ø and directory $. The drive and directory can therefore be omitted from file specifications. They will be assumed to have these values.

Typing

```
SAVE "MYPROG"
```

will automatically store your program in a file named

```
:Ø.$.MYPROG
```

assuming you have not changed the current drive and directory. (Chapter 6 'The filing system commands' explains how you can change the current drive and directory with the commands *DRIVE and *DIR.)

Multi-file operations

Another common term used to refer to multi-file operations is 'wild card' facilities. Some of the filing system commands can operate on a number of files instead of just one. These are all followed by the abbreviation <afsp> instead of <fsp> (<afsp> stands for 'ambiguous file specification'). *INFO is an example of such a command. It provides information about a named file, eg:

```
*INFO :Ø.$.MYPROG
```

RETURN

will display information about the file named MYPROG in directory $ on drive Ø.

However, it is possible that you want information about a number of files. The 'wild card' facilities enable you to specify several files for the command to operate on. The wild cards are provided by the characters * and # which have special meanings when they appear in the file specification, eg:

```
*INFO  :0.#.MYPROG
```

means: 'Display information about files called MYPROG in any directory on drive 0'.

```
*INFO  :0.$.MYPRO#
```

means: 'Display information about all files on drive 0 in directory $ with names starting "MYPRO" followed by any single character.' eg:

MYPROA, MYPROT and MYPROG and so on.

```
*INFO  ###        RETURN
```

gives information on all files with three-letter file names in the current directory.

The character * means multiple #s to the end of the field, eg

```
*INFO  :0.$.M*
```

will display information about any files on drive 0 and directory $ whose names begin with M.

```
*INFO  *A*        RETURN
```

gives information on all files in the current directory with an A in them: for example, A, AB, FREDA, PGRAM1 etc.

```
*INFO  *.*        RETURN
```

gives information on all files, in all directories.

Auto-start facilities

Sometimes it is useful to make a program or a file on one of your disks *LOAD, *RUN or *EXEC automatically when you insert the disk and press SHIFT and BREAK. This can be done using a file named !BOOT. !BOOT is a special file name recognized by the filing system when you start the computer by pressing SHIFT BREAK. If there is a file of specification

```
:0.$.  !BOOT
```

the filing system will do one of four things according to the option set on the disk using *OPT 4,n (see chapter 6).

Option 0: ignores !BOOT
Option 1: *LOADs !BOOT into memory
Option 2: *RUNs !BOOT as a machine-code program not a BASIC program
Option 3: *EXECs !BOOT

See chapter 6: 'The filing system commands' under the section *EXEC for an explanation of option 3. That section also describes how to use this auto-start facility to make the computer run one of your BASIC programs automatically.

The options can be changed using the *OPT 4 command. The 'Hello' program on the introductory and utilities disk is loaded using a !BOOT file.

As well as programs, you may wish to store data on the disks. The filing system provides facilities for storing and retrieving the data quickly and selectively under the control of your programs.

One of the methods is to use a type of file called a 'random access file' – see chapter 8.

Library directory

The Disk Filing System enables you to specify one drive/directory as the 'library'. This will always be set to :0.$ when you start the computer by pressing **BREAK**. It can be altered using the filing system command *LIB, until the next **BREAK**. All the utility programs should be located in the library. This is because when you type

*(utility name) **RETURN**

it is equivalent to typing

*RUN (utility name) **RETURN**

where the drive and directory are omitted and will be assumed to be either the current drive/directory or the library. The filing system will first search the current drive/directory for the file and then, if it cannot find it there, it searches the library.

5 The BREAK key

The **BREAK** key resets the BBC Microcomputer. However, the Disk Filing System can preserve some of its status after **BREAK**. There are two types of **BREAK** : the first is called 'hard break', and is achieved by holding down the **CTRL** key and pressing **BREAK** ; the second is called 'soft break', and is done by just pressing the **BREAK** key.

As far as the Disk Filing System is concerned, hard break is the same as switching the computer off and then on again. The directory and library are set to `:0.$` and any open files are forgotten about. Data written to a file which is still open may be lost. (Cold start.)

When you do a soft break, the Disk Filing System preserves its status, ie the directory and library remain the same, and open files remain open, but again – data may be lost. (Warm start.)

If you press **SHIFT** **BREAK** to make the Disk Filing System auto-start using the `!BOOT` file, the Disk Filing System does a cold start as per **CTRL** **BREAK**.

Important: To do a warm start, as per a soft break, or going `*TAPE` then `*DISK` to restart the DFS, the DFS must assume that the contents of RAM below the default value of PAGE have not been corrupted. This means that if you have been using proprietary software which uses all the RAM from &E00 upwards, you must exit the program with a hard break to obtain normal DFS operation.

6 The filing system commands

The Disk Filing System is an 8K byte program. BASIC programs are stored on a disk or tape, but the filing system is stored in Read Only Memory (ROM) inside the BBC Microcomputer. The filing system controls the reading and writing of information to and from the disks and provides a number of useful facilities for maintaining that information. The following pages describe all the filing system commands. They are words which the filing system program will recognize and act on. They can be typed directly on to the keyboard or embedded within your BASIC program. They are all prefixed with the * character which signals the computer that a filing system command follows. Each command is described under a number of sections with headings as follows:

Command
This is followed by a syntax abbreviation and a few words explaining the derivation of the word.

<drv> = drive
<fsp> = file specification
<dir> = directory
<afsp> = ambiguous file specification: this means that you can get the computer to perform an operation on a number of files at a time by typing in as much of the file names as is common to all these files – followed by a * character. For example, if you want to `COPY` the files called `FRED1`, `FRED2` and `FRED3` from drive 1 to 2, you could type

```
*COPY 0 2 FRED*          RETURN
```

Purpose
A plain English description of what the command does.

Example
This section gives a few one-line examples of the use of the commands. These examples are only intended to be illustrative.

Description
A description of the command using normal computer jargon.

Associated commands

This section lists commands which have similar functions or are normally used in conjunction with this command.

Demonstration program

Where it is needed, a short program is included to illustrate use of the filing system command in a BASIC program.

Notes

Particular points to watch for or special applications of the command are covered by additional notes if necessary.

Diagrams

Diagrams are used where they make the function of a command clearer.

*ACCESS <afsp> (L)

Purpose
To prevent a file from being deleted or overwritten. The command 'locks' or 'unlocks' a file. You cannot delete, overwrite or write to a locked file until you unlock it again. If you load a file which is locked, you will not be able to save it again with the same name. This is because saving a file with the same name as one already on the disk causes the one on the disk to be deleted and replaced with the new file. A locked file cannot be deleted.

Example

```
*ACCESS HELLO L
```

This locks the file HELLO.

```
*ACCESS HELLO
```

unlocks it again so that it can be deleted or overwritten.

Description
Sets or removes file protection on a named file. It prevents a number of other filing system commands from acting on the file.

Notes
Once locked, a file will not be affected by the following commands:

```
*SAVE
*DELETE
*WIPE
*RENAME
*DESTROY
```

If you attempt to use any of these commands on a locked file the message

```
Locked
```

is produced.

If you attempt to use ***ACCESS** on a write-protected disk the message

```
Disk read only
```

is produced.

Important: Locking a file does *not* prevent it from being removed from a disk with ***FORM80** or from being overwritten with ***BACKUP**.

*BACKUP (source drv) (dest. drv)

Purpose
To read all the information on one disk and write it to another, producing two disks with identical information.

Example

```
*ENABLE
*BACKUP 0 1
```

copies all the information on drive 0 onto drive 1.

Description
Sector by sector copy program.

Associated commands

```
*COPY
*ENABLE
```

Notes
*ENABLE must be typed before the command will work, otherwise the message

```
Not enabled
```

is displayed.

If you give 0 as the source and destination drives, eg:

```
*BACKUP 0 0
```

the program will alternately ask you to insert the source and destination disks into drive 0. This makes it possible to copy disks even if you only have a single drive.

Fig 4 in chapter 3 illustrates the process.

All information previously on the destination disk is overwritten, so be careful not to confuse the source and destination disks. If the source disk is blank, the destination disk will end up blank as well.

Warning: The contents of memory may be overwritten by this command. If you have a program or some data in memory that you want to keep, **SAVE** it before you use the command.

*BUILD <fsp>

Purpose
To create a file directly from the keyboard. After typing this command everything else entered will go into the named file. This is useful for creating EXEC files and the !BOOT file described in chapter 4.

Example

```
*BUILD !BOOT
```

will cause everything subsequently typed in to be entered into a file called !BOOT.

Line numbers are displayed on the screen to prompt you to enter your text as follows:

```
>*BUILD !BOOT
0001 FIRST LINE OF TEXT
0002 SECOND LINE
0003 ESCAPE
```

Pressing the **ESCAPE** key on a line by itself terminates a *BUILD command.

Description
Builds a file from the keyboard.

Associated commands

*EXEC
*LIST
*TYPE

*CAT (<drv>) catalog

Purpose

The command displays the catalog of a disk on the screen, showing all the files present on the disk. (drv) is the number of the drive you want displayed. If (drv) is omitted, the current drive is assumed.

Example

```
*CAT 0          RETURN
PROGRAM (nn)
Drive: 0                    Option: 2 (RUN)
Directory: 0.$               Library: 0.$

     !BOOT                       HELLO
     SUMS                        TABLE
     TEST                        VECTORS
     ZOMBIE

A. HELLO L                  B.SUMS
```

Note that the heading part of the catalog shows the drive number, the title of the disk, the currently set auto-start option of the disk (in this case 2 for RUN), and the currently selected library and directory. The files are displayed in alphabetical order reading across the two columns. In the example above there are nine files on the disk.

!BOOT to ZOMBIE are in the current directory $. The current directory's files are always listed first. A. HELLO is in directory A. It is also followed by L, meaning that it is a 'locked' file. (See *ACCESS for an explanation.) B.SUMS is in directory B and is not locked.

Description
Displays a disk catalog.

Associated commands

```
*INFO
*ACCESS
*TITLE
*OPT 4, n
*DIR
*DRIVE
```

*COMPACT <drv>

Purpose
Attempting to SAVE a program or file on to a disk may produce the message Disk full if there is no single space available on the disk big enough for the information. It may be that there is enough space, but it is split into several small sections. This command appends all spare space on a disk to the end. When you delete a number of files, the spaces they had occupied will probably be distributed over the disk with current files in between them. *COMPACT moves all current files to the 'start' of the disk leaving the space in one continuous block at the end.

Example

```
*COMPACT 1
$.HELLO  1700  801F  0003B  002
$.SUMS   1700  801F  00098  003
```

. . . and so on.

As 'compacting' proceeds, all the current files are displayed in the order in which they occur on the disk.

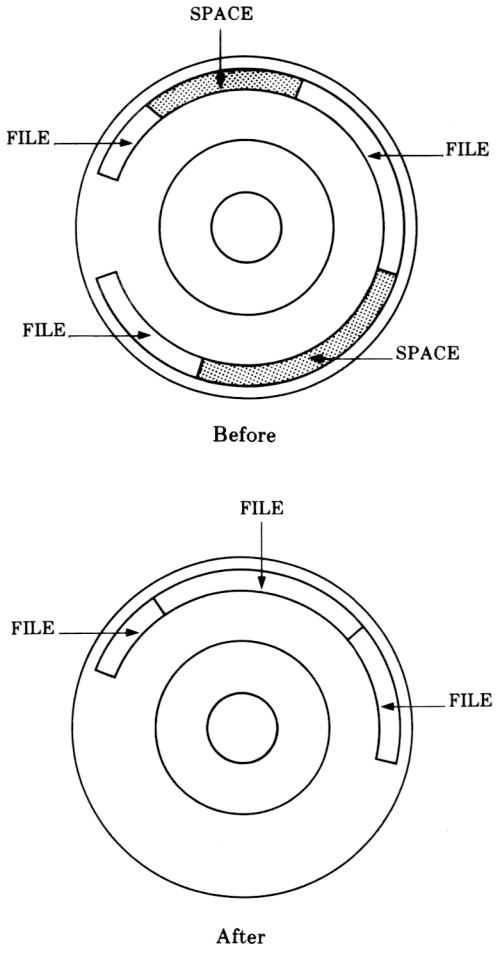
Description
Moves all available space on a disk into one continuous block following the current files.

Associated commands

*SAVE and BASIC's SAVE and OPENIN.

Notes
This facility will only do anything if there is space between the files. There will only be such space if a file has been deleted from between two others. Compacting a disk happens faster if you select mode 7 beforehand.

Fig 6 *COMPACT



Warning: This command may overwrite the contents of memory. If you have a program or data in memory that you want to keep, **SAVE** it before you use this command.

*COPY <source drv> <dest. drv> <afsp>

Purpose
To copy a named file or files from one disk to another.

Example

```
*COPY 0 1 HELLO
```

This copies a file called HELLO in the current directory on drive 0 onto drive 1.

Description
File copy program.

Associated commands

*BACKUP

Notes
The 'wild card' facilities may be used to specify a group of files to be copied, eg:

```
*COPY 0 1 #.MY*
```

Copies all files beginning MY irrespective of which directory they are in. Information already on the destination disk is not affected.

Diagram
Figs 3 and 4 in chapter 3 apply.

Warning: This command may overwrite the contents of memory. If you have a program or data in memory that you want to keep, SAVE it before you use this command.

*DELETE <fsp>

Purpose
To remove a single named file from the catalog of a disk. The space occupied by the file becomes available for other information. Succeeding file names in the catalog are shuffled up, but not the files themselves. Once a file is deleted you cannot get it back again.

Example

```
*DELETE FRED
```

removes a file called `FRED` from the current directory on the current disk.

Description
Single file deletion.

Associated commands

```
*WIPE
*DESTROY
*COMPACT
```

Notes
If the disk is write-protected the message

```
Disk read only
```

is produced. If the file is not found in the directory, the message

```
Not found
```

is displayed. If the file is locked, the message

```
Locked
```

appears.

Once deleted, a file cannot be restored.

*DESTROY <afsp>

Purpose
To remove specified files from the disk in a single action. This command may take the ambiguous file specification so that groups of files can be deleted. When you use this command, a list of the files to be deleted is displayed. A single Yes/No question appears at the end of the list offering you the choice to go ahead and delete all the listed files or not. Use this command with care, because its effect is not reversible. It will not attempt to remove locked files. (See *ACCESS.)

Example

```
*ENABLE
*DESTROY  *.H*
A.HELLO
$.HELLO

Delete  (Y/N)
```

If you type Y in reply to the question all the named files will be deleted. The message

```
Deleted
```

is displayed when the job is done. Typing anything else cancels the command.

Associated commands

```
*ENABLE
*WIPE
*DELETE
```

Notes
Once destroyed, files cannot be restored.

*ENABLE must be typed immediately before *DESTROY, or it will not work and the message

```
Not  enabled
```

is displayed.

*DIR (<dir>) Set the directory

Purpose
To change the current directory to (<dir>). After a cold start, the current directory is always set to $. To save files in a different directory in the catalog, you must use this command to change the current directory to the one you want and then SAVE them.

Example

```
*DIR  A
```

This sets the current directory to A. You now have access to any files in directory A in the catalog. Any files now saved using *SAVE, or BASIC's SAVE, will be in directory A.

Description
Sets the current directory to the argument supplied.

Notes
Directory can be set to any character except these four exceptions

```
#     *     .     :
```

This command does not alter the directories written in the catalog. It merely states which directory in the catalog you have access to by default.

You can change drive at the same time, and this takes the form

```
*DIR  :<drv>.<dir>
```

So if the current drive is drive 0, and the current directory is $, then

```
*DIR  :2.A
```

will set the drive to drive 2, and the directory to A. Alternatively, you can set the drive separately using the *DIR command:

```
*DIR  :2
```

*DRIVE <drv> Set current drive

Purpose
Changes the current drive to <drv>. Any commands which follow will work on <drv> until another drive is specified.

Example

```
*DRIVE 1
```

sets the current drive to 1 and

```
*CAT
```

will produce a catalog of drive 1.

```
*CAT 0
```

will catalog drive 0, but the current drive is still drive 1 until you change it back to 0, or after a cold start.

Description
Sets the current drive, and leaves the current directory unchanged.

*DUMP <fsp>

Purpose
Produces a hexadecimal listing of a file on the screen.

Example

```
*DUMP  SUMS
```

Description
Hexadecimal screen dump.

Associated commands

```
*LIST
*TYPE
```

Notes
It is useful to use this command in page mode so that the file is displayed one page at a time on the screen.

CTRL N selects page mode, CTRL O turns it off.

*ENABLE

Purpose
Some of the filing system commands produce irreversible effects. To prevent them from being used accidentally, it is necessary to type *ENABLE before they become operational. These commands are:

```
*BACKUP
*DESTROY
```

Example

```
*BACKUP
```

will not work; the message

```
Not enabled
```

is produced.

```
*ENABLE
*BACKUP
```

will work.

Notes
*ENABLE must be typed immediately before the command to be enabled. Any * name command typed in between nullifies the *ENABLE.

*EXEC <fsp> execute

Purpose
This command reads byte by byte all the information in a named file as if it was being typed on the keyboard. This is useful when you find that you are repeatedly typing the same sequence of commands. Instead you can build an EXEC file consisting of all these commands and type *EXEC <fsp> each time you want this sequence of commands. *BUILD <fsp> is an associated command used to create an EXEC file.

Example

```
*EXEC HELLO
```

takes the contents of file HELLO and reads it one character at a time as if it was being typed at the keyboard.

Description
Executes the contents of a named file by reading each byte as if it were coming from the keyboard.

Associated commands

*BUILD

Notes
One useful application of the *EXEC command is in association with the auto-start facilities described in chapter 4 and in the section on *OPT4 in this chapter. If you create a !BOOT file containing the BASIC keyword CHAIN followed by the file name of one of your BASIC programs, the effect of pressing SHIFT BREAK will be to load and run the BASIC program automatically.

*HELP <keyword>

Purpose

Displays useful information on the screen. In the disk system this consists of a list of the filing system commands or the utilities depending on the <keyword> used. The two keywords which produce a response in the Disk Filing System are UTILS and DFS. If just *HELP is typed, the system produces a list of currently installed ROMs and a list of the keywords which will produce further information.

Example

```
*HELP

DFS 1.00
   DFS
   UTILS

OS 1.20

*HELP DFS
DFS 1.00

   ACCESS <afsp> (L)
   BACKUP <source> <dest.>
   COMPACT (<drive>)
   COPY <source> <dest.> <afsp>
   DELETE <fsp>
   DESTROY <afsp>
   DIR (<dir>)
   DRIVE <drive>
   ENABLE
   INFO <afsp>
   LIB (<dir>)
   RENAME <old fsp> <new fsp>
   TITLE <title>
   WIPE <afsp>

OS 1.20
```

```
*HELP UTILS

:DFS 1.00
    BUILD <fsp>
    DISK
    DUMP <fsp>
    LIST <fsp>
    TYPE <fsp>
```

Notes

*RUN, *SPOOL, *SAVE, *EXEC and *LOAD are not included in these lists because they are Machine Operating System commands which operate outside the Disk Filing System. *HELP is a Machine Operating System command.

*INFO <afsp>

Purpose
Displays information about a file or group of files. It includes details not given by *CAT such as the length of the file and its location. It is displayed in the following order across the screen.

Directory	File name	Access	Load address	Execution address	Length in bytes	Start sector

Example

```
*INFO A. HELLO
```

displays

```
A.     HELLO L   001900 00801F 00003B 003
```

Description
Displays detailed file information.

Associated commands

*CAT

Notes
If the file is not found on the specified (or assumed) drive and directory the message

```
Not found
```

is produced. The command must be re-entered using the correct <afsp>.

The wild card facilities # and * may be used if you want information about a group of files.

*LIB :(<drv>). <dir> Selects the library

Purpose
Sets the library to the specified drive and directory.

Example

```
*LIB :1.A
```

sets the library to drive 1 directory A. After this typing

```
*<filename>
```

will search directory A on drive 1 for the named file and if it is found the file will be loaded and executed just as if you had typed

```
*RUN :1.A. <filename>
```

Description
Sets the drive/directory containing the library.

Associated commands

```
*RUN
```

Notes
The library can contain files which are utility programs, designed to act on other files eg sorts, edits and merges are all common utility programs. It is then possible to say:

```
*SORT FRED
```

where SORT is the name of the file in the library and FRED is the name of another file in the current drive and directory. This makes use of the fact that any text after the <fsp> is stored in memory and is available to your machine code program for interpretation. A pointer to the start address of this text is available to your program via a call to OSARGS with Y=Ø, A=1 and X=the address of the byte block in page Ø where the text is stored. To read the text stored at this location you must use a call to OSWORD with A=5.

OSWORD call with A=5

Read I/O processor memory. This call enables any program to read a byte in the I/O processor no matter in which processor the program is executing. On entry, X and Y point to a block of memory as follows:

XY LSB of address to be read
XY+1
XY+2
XY+3 MSB of address to be read

On exit, the eight-bit byte will be stored in XY+4.

As this routine reads one byte at a time, you may need to use repeated calls to it to recover all the text following the <fsp>.

Chapter 9 provides more information about using the disk system from assembler programs.

*LIST <fsp>

Purpose
Displays a text file on the screen with line numbers.

Example

```
*LIST  DATA
```

displays the contents of the file called `DATA` on the screen, line by line with each line numbered.

Description
ASCII list with line numbers.

Associated commands

```
*TYPE
*DUMP
```

Notes
BASIC is 'tokenized', so listing a BASIC program file will display nonsense. (See the BBC Microcomputer User Guide). An ASCII text file of a BASIC program can be obtained using the `*SPOOL` command.

Files written with BASIC keyword `PRINT#` can also be listed with this command.

In page mode the listing will stop after displaying each screenful, until you press either **SHIFT** key to make it continue. **CTRL** `N` turns page mode on, **CTRL** `O` turns it off.

*LOAD <fsp> <address>

Purpose
Reads a named file from the disk into memory in the computer starting at either a specified start address or the file's own load address.

Example

```
*LOAD  HELLO
```

Reads the file HELLO into memory starting at location 19ØØ (hex), which is the load address of the file when it was saved.

(See example in *INFO.)

```
*LOAD  HELLO  32ØØ
```

Reads the file HELLO into memory starting at location 32ØØ (hex).

Other examples are

```
*LOAD  "HELLO"
*LOAD  "HELLO"  32ØØ
```

Description
Loads a file into memory.

Associated commands

```
*SAVE
*RUN
```

Notes
All the above are valid commands. The quotation marks are optional; but either a pair, or none should be present. The named file must be in the current directory on the current disk. If the file is not found, the message

```
Not  found
```

is produced.

*OPT 1 (n)

Purpose
This command enables or disables a message system which displays a file's information (the same as *INFO). Every time a file on the disk is accessed, the information is displayed. (n) can be any number greater than Ø to enable the feature. (n) = Ø disables it.

Example

```
*OPT 1 1 or *OPT 1, 1
```

enables the messages;

```
*OPT 1 Ø or *OPT 1, Ø
```

disables the messages.

Description
Message system to display file information at every access.

Associated commands

```
*INFO
```

Notes
A space or a comma between *OPT 1 and its argument (n) is essential.

*OPT 4 (n)

Purpose
Changes the auto-start option of the disk in the currently selected drive. There are four options to choose from: Ø, 1, 2 or 3. Each option initiates a different action when you press **SHIFT** and **BREAK** on the computer. The computer will either ignore or automatically *LOAD, *RUN or *EXEC a file called !BOOT which must be in the directory $ on drive Ø.

Example

```
*OPT  4  Ø   does nothing
*OPT  4  1   will *LOAD the file !BOOT (ie a machine-code file)
*OPT  4  2   will *RUN the file !BOOT (ie a machine-code file)
*OPT  4  3   will *EXEC the file !BOOT (ie a BASIC file)
```

When *OPT 4 2 is set, and a machine-code program (ie !BOOT) is executed, it is important to remember that the interrupt flag is undefined (unlike *RUNning the program, where the interrupt flag is cleared). It is suggested that your machine-code !BOOT files contain a CLI to clear the interrupt flag.

Description
Changes the start-up option of a disk.

Notes
It is essential to include a space between the command and (n). *OPT4Ø would produce the message

```
Bad option
```

If the disk is write-protected the error message

```
Disk read only
```

is produced in response to the *OPT 4 command.

If the option Ø is set, the !BOOT file need not be there. With any other option the message

```
Not found or File not found
```

is produced if !BOOT is not found in directory $ on drive Ø.

Important: Do not confuse *OPT 4 with BASIC keyword OPT or *OPT 1. They are completely different. Refer also to chapter 12 where it describes how to swap the effects of **BREAK** and **SHIFT** **BREAK**.

The utilities disk is set to option 2 when you receive it so when you press **SHIFT** **BREAK** the 'HELLO' program, saved as file !BOOT, is *RUN automatically.

*RENAME (old fsp) (new fsp)

Purpose
Changes the file name and moves it to another directory if required.

Example

```
*RENAME  SUMS  B.MATHS
```

Assuming that the current directory is $, the file **$. SUMS** becomes **B.MATHS**.

Description
Renames a file.

Notes

```
*RENAME   :0.$.SUMS   :  1.B.MATHS
```

is not allowed. The file cannot be moved from drive Ø to drive 1 using *RENAME. Only the directory and file name can be changed.

If the file does not exist the message

```
Not  found
```

is displayed. If the first file is locked

```
Locked
```

is displayed. If the disk is write-protected

```
Disk  read  only
```

is displayed. If the <new fsp> has already been used the message

```
Exists
```

is displayed.

*RUN <fsp> (parameters to utility)

Purpose
This command is used to run machine-code programs. It loads a file into memory and then jumps to the execution address of that file.

Example

```
*RUN   PROG
```

will cause a machine-code program in the file called PROG to be loaded and executed starting at the execution address of the file.

Description
Runs a machine-code program.

Associated commands

*SAVE
*LIB (for an explanation of 'parameters to utility')

Notes
This command will not run a BASIC program.

Typing *<fsp> is accepted as being *RUN <fsp>.

Typing *<filename> results in the file being loaded and executed if it is found in the currently selected drive/directory or the library.

*SAVE <filename> <start address> <finish address> (<execute addr.>) (<reload addr.>)

Purpose
It is important not to confuse this with the BASIC keyword SAVE – they are quite different. This command takes a copy of a specified section of the computer's memory and writes it on to the disk in the current drive/directory. It is put into a file of the given name. You will mostly use this command to record your machine-code programs.

Example

```
*SAVE "PROG"  SSSS FFFF (EEEE) (RRRR)
*SAVE "PROG"  SSSS + LLLL (EEEE) (RRRR)
```

SSSS = Start address of memory to be saved
FFFF = Finish address
EEEE = Execution address (see below)
RRRR = Reload address
LLLL = Length of information

Notes
RRRR and EEEE may be omitted in which case the reload address and the execution address are assumed to be the same as the start address.

If the disk is write-protected the message

```
Disk read only
```

is produced. If there are already 31 files on the disk the message

```
Cat full
```

is produced. If the specified file name already exists and is locked the message

```
Locked
```

is produced. If the file already exists but is unlocked it is deleted. Then starting in sector 2 track Ø a gap large enough to hold the new information is searched for. If none is found the message

```
Disk full
```

is produced.

If enough space is available the information is written on to the disk and the file name is entered on to the catalog in the current directory.

*SPOOL <fsp>

Purpose
Prepares a file of the specified name on the disk to receive all the information subsequently displayed on the screen. This is a very useful command particularly for producing a text file of one of your BASIC programs. (See notes below.)

Example
You can obtain a text file of one of your BASIC programs as follows:

```
LOAD   "MYPROG"
```

Loads a program from disk into memory.

```
*SPOOL  FRED
```

Opens a file called `FRED` on the disk ready to receive information from the screen.

```
LIST
```

Causes the BASIC program to be displayed on the screen and also written onto the file called `FRED`.

```
*SPOOL
```

Turns off the 'spooling' and closes the file called `FRED`.

Description
Spools subsequent output to the screen to a named file opened for the purpose. Closes the file when spooling is terminated.

Associated commands

*BUILD
*EXEC
*LIST
*TYPE

Notes

BASIC on the BBC Microcomputer is 'tokenized'. This means that the lines which you type in your program are abbreviated inside the computer's memory and on the disk. A program file will contain these abbreviated 'tokens' rather than your original program text.

*TITLE <disk name>

Purpose
Changes the title of the disk in the current drive to the first 12 characters after the command. It fills in with spaces if there are less than 12 characters. Any characters are allowed.

Example

```
*TITLE "MY DISK"
```

This sets the title to `"MY DISK"` with five spaces added on the end.

```
*TITLE "A DIFFERENT TITLE"
```

This changes the title to `A DIFFERENT`. Anything after the first 12 characters is ignored. The quotation marks are only required if the title includes spaces.

Notes
If the disk is write-protected the message

```
Disk read only
```

appears when you try to use this command.

*TYPE <fsp>

Purpose
Displays a text file on the screen without line numbers.

Example

```
*TYPE HELLO
```

Description
Screen list of a named file.

Associated commands

```
*LIST
*DUMP
```

Notes
BASIC programs are not stored on disk as text files when you SAVE them, so this command will display nonsense if you try to display a BASIC program.

Page mode is selected with CTRL N and turned off by CTRL O. Page mode allows you to see a screenful of text at a time: when the screen has filled up, instead of immediately scrolling, the computer waits until you press the SHIFT key. The next screenful then appears.

*WIPE <fsp>

Purpose
Removes specified files from the catalog and rearranges the catalog. Asks for confirmation that each file conforming to the specification is to be deleted.

Example

```
*WIPE  *.SU*
```

is a request to delete all files on the current drive beginning with the letters SU. As each file is found the file name is displayed like this:

```
A.SUM
```

At this point only type Y if you want to delete the file. Typing anything else leaves the file intact.

Description
Delete with <afsp> and confirmation per file.

Associated commands

```
*DESTROY
*DELETE
```

Notes
Once deleted using *WIPE, a file cannot be restored. Locked files are not removed. (See *ACCESS.)

7 The filing system utilities

As explained in chapter 6 the filing system commands are, in fact, programs stored in ROM (Read Only Memory) inside the BBC Microcomputer.

The utilities are similar programs but they are stored on the introductory and utilities disk. They are used in the same way as the commands by typing

*(utility name)

The introductory and utilities disk must be in the disk unit at the time. The utility must either be in the current directory on the current drive or in the library (see command *LIB).

There is one 'formatting' utility supplied which is for 80 track disks. On the introductory and utilities disk this is normally in drive 0, directory $, which are the default values on start-up for both the library and the current drive/directory. (See commands *DRIVE and *DIR.) The second utility is *VERIFY.

The following pages describe the utilities. You can add utilities of your own by saving machine-code programs into the library.

*FORM8Ø (<drv>)

Format an 8Ø track disk

Purpose
The command prepares new disks for use with the filing system on the BBC Microcomputer. It marks areas of disk where information will be stored and sets up a catalog. The catalog is empty at first, but when you store programs and data on the disk, the catalog records their positions on the disk. They can then be retrieved quickly by reference to the catalog. While formatting the information put on to the disk is verified automatically.

Example
To format a new disk insert the introductory and utilities disk into the disk drive and type

```
*FORM8Ø
```

The computer asks

```
Format which drive?
```

Type the number of the drive (Ø to 3) you wish to format, eg Ø. The computer then asks

```
Do you really want to format drive Ø?
```

Remove the introductory and utilities disk. Insert the disk to be formatted and then answer

```
Y
```

The formatting starts immediately and the message

```
Formatting drive (n)
```

is displayed. As each track is formatted and verified the track number is displayed as follows:

```
00 01 02 03 04 05 06 07 08 09
0A 0B 0C 0D 0E 0F 10 11 12 13
14 15 16 17 18 19 1A 1B 1C 1D
1E 1F 20 21 22 23 24 25 26 27
28 29 2A 2B 2C 2D 2E 2F 30 31
32 33 34 35 36 37 38 39 3A 3B
3C 3D 3E 3F 40 41 42 43 44 45
46 47 48 49 4A 4B 4C 4D 4E 4F
disk formatted
```

Above is the response if the formatting was successful. If a ? appears in this list, it means that a retry occurred, so you're advised to reformat the disk. If the formatting was not successful, then either the message

```
Verify error
```

or

```
Format error
```

is displayed.

After successful formatting the message

```
Repeat format (Y/N)
```

is displayed. You can format another disk straight away.

Description
Initializes disks with track and sector format. Clears the catalog and verifies the sectoring.

Associated commands

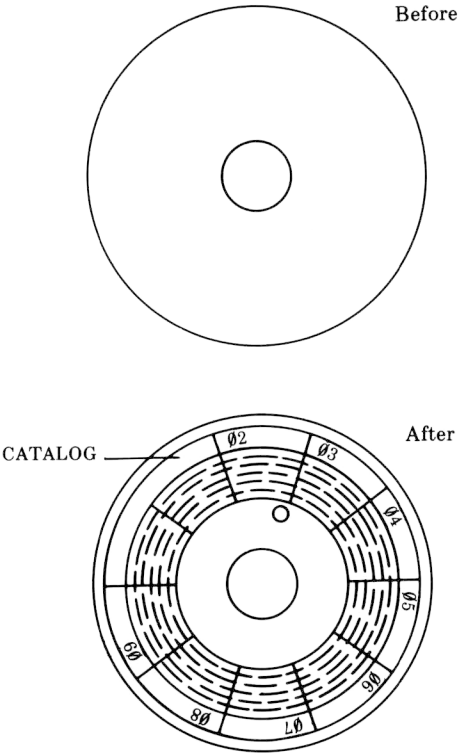
```
*VERIFY
```

Notes

If you find that verification or formatting fails persistently it could be that either the disks you have are poor quality or that the disk drive needs servicing.

If you have a dual drive system you can format a disk in the second drive while the introductory and utilities disk remains in the first drive.

Fig 7 Formatting



*VERIFY (<drv>)

Purpose
Checks each sector of a disk for legibility. It is done automatically when you use the *FORM80 command.

Example

```
*VERIFY 1
```

verifies drive 1.

Description
Sector verification program.

Associated commands

*FORM80

8 Random access files

One of the major advantages of a disk over a cassette tape is that the read/write head of the disk drive can be moved to a specific place on the disk quickly and accurately. Imagine you have a data file on cassette tape consisting of 'Names' and 'Telephone numbers'. To find a specific telephone number the file must be loaded and read from the beginning until the required record is found. If the file is long this will take some time. On the other hand, the Disk Filing System allows you to move to the required record and just read that one. Clearly this is much quicker.

To make this possible the Disk Filing System provides a pointer. The pointer points to a particular character in the file. It is the next character on the file to be read or written. In BASIC the pointer is controlled by the keyword `PTR#`. The other keywords in BASIC which are used in connection with disk files are `EXT#` and `EOF#`. `EXT#` tells you how long a file is, `EOF#` returns a value of `TRUE (-1)` if the end of file has been reached and `FALSE (0)` if not. All the BASIC keywords used to manipulate disk files are explained in the User Guide. They are:

```
OPENOUT
OPENIN
OPENUP
PTR#
EXT#
INPUT#
PRINT#
BGET#
CLOSE#
BPUT#
EOF#
```

To prepare a file to receive data the `OPENOUT` keyword is used. In the User Guide the following example is given:

```
330  X=OPENOUT ("cinemas")
```

The effect of this line in a BASIC program is as follows:

1. If a file called 'cinemas' exists it is deleted.

2. A file called 'cinemas' is entered on to the disk catalog of the currently selected drive, in the current directory.

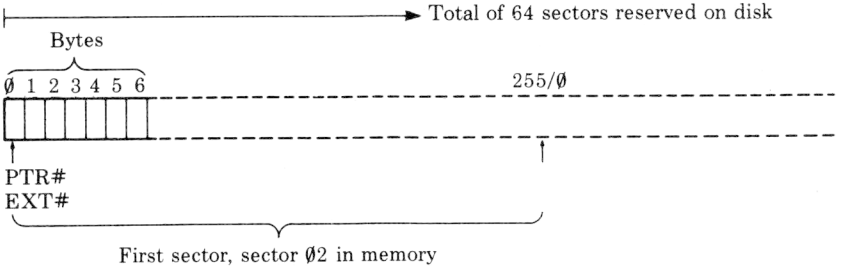
3. The filing system reserves 64 sectors (or the length of the previous file called 'cinemas' if there was one) on the disk for the exclusive use of the file 'cinemas'. If 64 sectors are not available, the file is not created and an error is produced.

4. Evaluating PTR# and EXT# at this point will reveal that they are both set to zero.

5. The filing system will have loaded into memory the first sector, 256 bytes of the file. This area of memory is specially reserved by the filing system for this purpose and is referred to as the 'buffer'.

Notice that the first action of the keyword OPENOUT is to delete any existing file of the specified name.

If there were no files on the disk previously, the effect can be illustrated as follows:

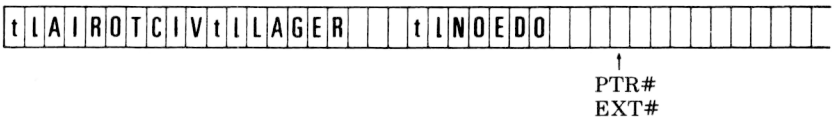


Nothing has been written on the file, so the value of EXT# (extent) is zero.

We can now use the BASIC keyword PRINT# to write three cinema names into these slots of ten characters each, as follows:

```
340  A = PTR#X
350  PRINT# X, "VICTORIA"
360  PTR#X = A+10
370  PRINT# X,"REGAL"
380  PTR# X = A+20
390  PRINT# X, "ODEON"
400  PTR# X = A+30
```

In practice you can do it much more elegantly than shown above. Nevertheless the result immediately after line 400 is:



Notice that the cinema names (in this illustration, VICTORIA) are in the file backwards. They are preceded by two bytes, represented in the diagram by 't' and 'l' . 't' specifies the type of data which follows. In this case the type is 'string' so the first byte will contain &00 in hex, as indicated below.

't' = &00 = String type, followed by 'l', followed by the string.

't' = &40 = Integer type, followed by four bytes containing the integer.

't' = &FF = Real type, followed by five bytes containing the real number.

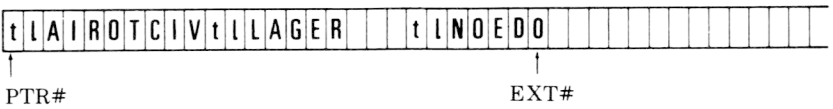
In our example the second byte, represented by 'l', gives the length of the string in hex. The integer and real number types are of fixed length as indicated above so they do not require the byte represented by 'l' to give the length. Real numbers are stored in exponential format, integers are stored with the high order bytes first in the file.

In the example we have used only the first 26 bytes of the file, so everything written to the file fits into the first sector which is in a 'buffer' in memory. If we had gone on writing names, the filing system would eventually have put the information in the memory buffer on to sector 02 of the disk and loaded sector 03 into the buffer to continue. This is still assuming that there are no other files on the disk, otherwise different sectors would be used. Remember that sectors 00 and 01 are reserved by the Disk Filing System for the disk catalog. Clearly then, at the end of a sequence of writing actions, we are left with a buffer in memory which may be partly filled with information. We must make sure that this information is written to the disk. This is done with the `CLOSE#` keyword in BASIC. This empties the buffer and frees the channel on which we opened the file (X in the example).

We can now read the information back from the disk if we want to. **OPENIN** is the BASIC keyword used to do this, eg:

```
5   DIM  cine$(3)
10  X  =  OPENIN  ("cinemas")
20  B  =  1
30  FOR  A  =  0  TO  20  STEP  10
40  PTR#X  =  A
50  INPUT#X,cine$(B)
60  B  =  B+1
70  NEXT  A
```

Line 10 of the example opens the file 'cinemas', loads the first sector into the buffer and sets **PTR#** to zero and **EXT#** to the length of the file.



Lines 30 to 50 of the example read each cinema name into an element of the array **cine$**, advancing the pointer to the start of the next name after reading each one. Now you can see why we stored each name in its own '10-byte record'. This makes it much easier to write a program to find the names again.

The important principle about using random access files is that you must keep track of where each item of information is written. You can then set **PTR#** to point to it again when you want to read or change it. The examples illustrate the basis of a very simple technique. There are a number of others which you can devise.

Note 1: As shown earlier in this discussion, **OPENOUT** reserves 64 sectors for a file. Other files opened may reserve sectors which immediately follow, eg:

```
X  =  OPENOUT  ("cinemas")
Y  =  OPENOUT  ("clubs")
```

The statements reserve 128 sectors consecutively if the disk was otherwise empty. It may be that you require more than 64 sectors for the first

file 'cinemas'. If so, you will need to write 'dummy' records to the file to extend it to the required length before you open the second file, eg:

```
10  X = OPENOUT ("cinemas")
20  FOR   A = 1 TO 200
30  PRINT#X, "DUMMY NAME FIELD"
40  PRINT#X, "DUMMY ADDRESS LINE ONE"
50  PRINT#X, "DUMMY ADDRESS LINE TWO"
60  PRINT#X, "ADDRESS LINE THREE"
70  PRINT#X, "POST CODE 123"
80  NEXT A
```

This program creates a file 79 sectors long with the dummy name and address written every 100 bytes.

By writing beyond the reserved area in this way you can effectively reserve as many sectors as you like. You can then open other files in the remaining space on the disk. `EXT#` will give the position of the '3' after the last dummy address on the file (20000).

The above method will only work consistently if you start with a blank, formatted disk. If you want to create a random access file larger than 64 sectors on a disk with other files already on it, there is another method.

First save the file with the name you want and with the number of bytes required. Use the address parameters of the `*SAVE` command to specify the number of bytes, eg:

```
*SAVE "DATA" 00000 08000
```

will create a file of 128 sectors (32K) called `DATA`. You can then open the file later in your program. The file will contain miscellaneous data which you can overwrite with the information you actually want. This second method causes the filing system to search the disk for a free space large enough to hold the file. Existing files will be skipped over if they would otherwise overlap with the new file.

Note 2: Up to five files may be open at any one time. This is because the space reserved for each file in the computer's memory to hold the information about extent, pointer etc is limited. In certain versions of the Disk Filing System, the commands `*LOAD`, `*SAVE`, `*EXEC`, `*SPOOL`, `*BUILD`, `*LIST` and `*DUMP` each use the space occupied by the information relating to one open file while they are active.

9 Using the filing system in assembler

Chapter 43 of the new User Guide is essential reading for anyone wanting to write assembler programs on the BBC Microcomputer. Most of the necessary information for using the filing system in assembler is presented there. In this chapter the main points are summarized and a particular use of OSWORD is described in detail.

General principles

There are a number of routines available to handle disk I/O. All the routines must be called with a JSR and the decimal flag clear. It is important that you use these routines. They are called in address range &FFØØ to &FFFF. They then call an internal (ROM resident) routine whose address is stored in RAM between &Ø2ØØ and &Ø2FF. The address here will vary according to the filing system in use. For example, the routine OSFIND to open or close a file is entered at &FFCE. It is indirected via &Ø21C. &Ø21C and &Ø21D contain the address of the executable routine in the Disk Filing System ROM. You can intercept the call by changing the addresses in these RAM locations.

Using the available routines you can perform all necessary functions relating to disk files. The relevant routines together with their entry points are as follows:

OSFIND	&FFCE	FINDV	&Ø21C	Open or close a file
OSARGS	&FFDA	ARGSV	&Ø214	Load or save data about a file
OSFILE	&FFDD	FILEV	&Ø212	Load or save a complete file
OSBGET	&FFD7	BGETV	&Ø216	Read a single byte to A from the file
OSBPUT	&FFD4	BPUTV	&Ø218	Write a single byte from A to the file
OSGBPB	&FFD1	GBPBV	&Ø21A	Load or save a number of bytes
OSWORD	&FFF1	WORDV	&Ø2ØC	With A=&7F and a parameter block, loads or saves a sector

OSFIND

Opens a file for writing or reading and writing. The routine is entered at &FFCE and indirects via &21C. The value in A determines the type of operation.

A=Ø	Causes a file or files to be closed
A=&4Ø	Causes a file to be opened for input (reading)
A=&8Ø	Causes a file to be opened for output (writing)
A=&CØ	Causes a file to be opened for input and output (random access)

If A=&4Ø, &8Ø or &CØ the Y (high byte) and X (low byte) must contain the address of locations in memory which contain the file name terminated with carriage return (&ØD). On exit A will contain the channel number allocated to the file for all future operations. If A=Ø on exit, then the operating system was unable to open the file.

If A=Ø then a file, or all files, will be closed depending on the value of Y. Y=Ø will close all files, otherwise the file whose channel number is given in Y will be closed.

On exit C, N, V and Z are undefined and D=Ø. The interrupt state is preserved, but interrupts may be enabled during the operation.

OSARGS

This routine enables a file's attributes to be read from file or written to file. The routine is entered at &FFDA and indirects via &214. On entry, X must point to four locations in zero page and Y contains the channel number.

If Y is non-zero, then A will determine the function to be carried out on the file whose channel number is in Y.

A = 1	Write sequential pointer
A = Ø	Read sequential pointer
A = 2	Read length
A = &FF	'Ensure' that this file is up to date on the media

If Y is zero then the contents of A will determine the function to be carried out.

A=Ø will return, in A, the type of file system in use. The value of A on exit has the following significance.

Ø No file system
1 12ØØ baud cassette file system
2 3ØØ baud cassette file system
3 ROM pack file system
4 Disk file system
5 Econet file system
6 Teletext/Prestel Telesoftware file system

A=1 will return the address of the rest of the command line in the zero page locations.

A=&FF will 'ensure' that all open files are up to date on the media.

On exit
X and Y are preserved, C, N, V and Z are undefined and D=Ø. The interrupt state is preserved, but interrupts may be enabled during the operation.

OSFILE

This routine, by itself, allows a whole file to be loaded or saved. The routine is entered at &FFDD and indirects via &212.

On entry
A indicates the function to be performed. X and Y point to an 18-byte control block anywhere in memory. X contains the low byte of the control block address and Y the high byte. The control block is structured as follows from the base address given by X and Y.

OSFILE control block

ØØ Ø1	Address of file name, which must be terminated by &ØD	LSB MSB
Ø2 Ø3 Ø4 Ø5	Load address of file	LSB MSB

Ø6 Ø7 Ø8 Ø9	Execution address of file	LSB MSB
ØA ØB ØC ØD	Start address of data for write operations or length of file for read operations	LSB MSB
ØE ØF 1Ø 11	End address of data, ie byte after last byte of data to be written, or file attributes	LSB MSB

The following table indicates the function performed by OSFILE for each value of A.

A=Ø	Save a section of memory as a named file. The file's catalog information is also written
A=1	Write the catalog information for the named file
A=2	Write the load address (only) for the named file
A=3	Write the execution address (only) for the named file
A=4	Write the attributes (only) for the named file
A=5	Read the named file's catalog information; place the file type in A
A=6	Delete the named file
A=&FF	Load the named file

When loading a file, the byte at XY+6 (the LSB of the execution address) determines where the file will be loaded in memory. If it is zero then the file will be loaded to the address given in the control block. If non-zero then the file will be loaded to the address stored with the file when it was created.

The file attributes are stored in four bytes. The least significant eight bits have the following meanings:

Bit	**Meaning**
Ø	Not readable by you
1	Not writable by you

2	Not executable by you
3	Not deletable by you
4	Not readable by others
5	Not writable by others
6	Not executable by others
7	Not deletable by others

File types are as follows:

Ø	Nothing found
1	File found
2	Directory found

A BRK will occur in the event of an error, and this can be trapped if required.

On exit X and Y are preserved, C, N, V and Z are undefined and D=Ø. The interrupt state is preserved, but interrupts may be enabled during the operation.

Read/write one byte

OSBGET call address &ØFFD7 to get a byte.
OSBPUT call address &ØFFD4 to put a byte.

Y contains the channel number on which the file was opened using OSFIND.
X is not used, but preserved.
A contains the byte to be put or receives the byte which is read.

The position in the file where the action of the GET or PUT takes place is determined by the position of the pointer as set by OSARGS.

On exit
C clear implies a successfully completed transfer.
C set implies end of file reached before completion of transfer.

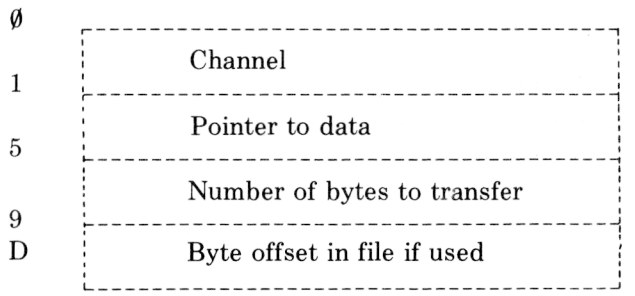
Read/write a group of bytes

OSGBPB call address &ØFFD1

This routine will write or read a byte (or group of bytes) to or from a specified open file. The option to read or write is determined by the value of A. The length of the data and its location are specified in a control block in memory.

On entry
X (low byte) and Y (high byte) point to the instruction block:

Offset



A determines the type of operation:

A = &Ø1 Put byte using byte offset
A = &Ø2 Put byte ignoring byte offset
A = &Ø3 Get byte using byte offset
A = &Ø4 Get byte ignoring byte offset
A = &Ø5 Read title, option, drive
A = &Ø6 Read current directory
A = &Ø7 Read current library
A = &Ø8 Read file names

This method is particularly useful in the environment of the Econet® file system, where the 'packaging overhead' for transferring small amounts of data is proportionately high.

On exit
A = Ø implies operation attempted.
A unchanged implies filing system does not support this facility.
C clear implies a successfully completed transfer.
C set implies end of file reached before completion of transfer. The number of bytes and byte offset (if used) are modified to show how much data has been transferred (usually as much as was asked for) and the new pointer value is the old pointer plus the amount transferred.

A = &Ø1 to &Ø4
The number of bytes is modified to show how much data has not been transferred (usually zero). The pointer value is updated (ie old pointer

plus the amount transferred), and the offset is set to its current value.

A = &Ø5
This returns in the area pointed at by the pointer the title, option and drive of the currently selected disk. It is returned in this form:

<title length><title><option><drive>

The cycle number, option and drive are single binary bytes.

A = &Ø6
This returns the currently selected directory. It is returned in this form:

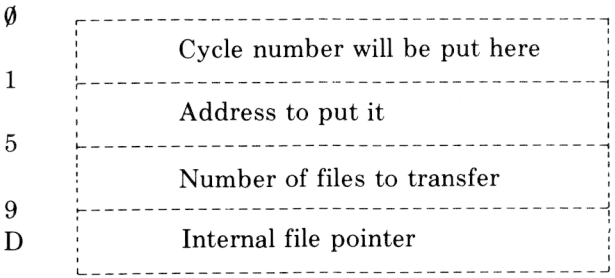
<length of disk name><disk name><length of directory name>
<directory name>

A = &Ø7
This returns the currently selected library. It is returned in this form:

<length of disk name><disk name><length of library name>
<library name>

A = &Ø8
This returns the files in the current directory. The format of the control block is similar to that for sequential files:

Offset



If the pointer is set to zero, the search will begin with the first file. All are updated in a similar manner to the way pointers are updated for A=&Ø1 to &Ø4. The format of the file names is as follows:

<length of filename 1><filename 1><length of filename 2>
<filename 2> . . .

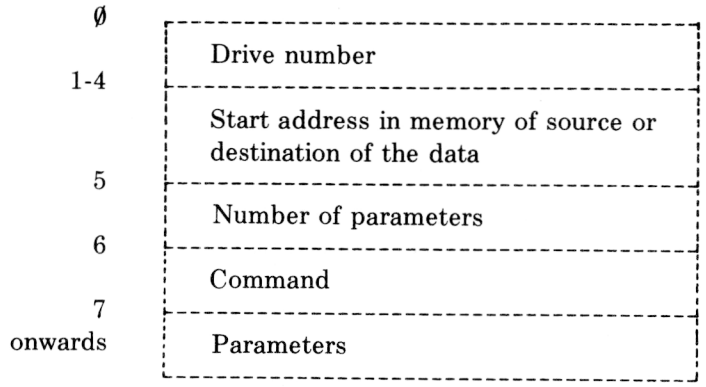
Read/write a sector

OSWORD with A =&7F
Call address at &FFF1

A=&7F indicates that a general read/write operation is required.

On entry X (low byte) and Y (high byte) point to the instruction block:

Offset



Example:

Number of parameters = 3

Command =&53 to read or &4B to write

Parameter 1	=	Track number
Parameter 2	=	Sector number
Parameter 3	=	&21 (specifies sector length of 256 bytes and 1 to be acted upon)

On exit
Ø in the last parameter address +1 indicates a successful transfer. A failure is indicated by a disk error number.

1Ø Changing filing systems

Your computer can have several filing systems available other than the Disk Filing System. The following commands are all used to exit from the current filing system into the one named.

*TAPE3	3ØØ baud cassette
*TAPE12	12ØØ baud cassette
*TAPE	12ØØ baud cassette
*NET	Econet filing system
*TELESOFT	The Prestel and Teletext system
*ROM	The cartridge ROM system
*DISK	Enters the filing system from one of the others
*DISC	Alternative spelling for above

Typing the command to enter the system you are already in has no effect.

11 Error messages

&BD Not enabled
An attempt is made to use a restricted command without typing *ENABLE immediately preceding it.

&BE Cat full
The catalog has enough space for 31 files. This error is produced if you attempt to enter more than 31.

&BF Can't extend
An attempt is made to extend a random access file when there is insufficient space immediately after it to do so.

&CØ Too many open
This error occurs on attempting to open a sixth random access file. Five is the maximum allowed at once.

&C1 Read only
This error occurs if you try to write to a file opened for reading only, using OSFIND with A=&4Ø.

&C2 Open
This error occurs if you attempt to open a file which is already open. An intervening CLOSE is required between two OPENs referring to the same file. This error is also produced if you try to delete an open file with the commands: *DELETE, *SAVE, *BUILD etc.

&C3 Locked
Access to the named file is locked. This error message is displayed when any attempt is made to overwrite or write to a locked file.

&C4 Exists
This occurs if you try to rename a file with an existing file name.

&C5 Drive fault
This error means that the disk drive is probably faulty and needs attention.

&C6 Disk full
This indicates that there is not enough space on the disk to open (OPENOUT#) or save a file of the specified size.

&C7 Disk fault
If this error message occurs it means that the computer cannot read the disk. It implies that the disk is damaged, faulty, unformatted or of the wrong type, like an 80 track disk in a 40 track drive.

&C8 Disk changed
This error occurs if the computer detects that a disk has been changed while files on it are still open.

&C9 Disk read only
This error occurs if you attempt to write to a disk which has the write-protection notch covered.

&CB Bad option
There are currently two 'option' commands *OPT1 and *OPT4. The error occurs if you type anything else besides 1 and 4 after *OPT.

&CC Bad name
This message appears if you enter a file name which is invalid, such as: longer than seven characters, or a blank file name, or control characters.

&CD Bad drive
This error means that the :(drv) part of the file specification was incorrect, eg ':' colon missing or drive number out of range 0 to 3.

&CE Bad dir
Means that the specified directory is not allowed, eg more than one character.

&CF Bad attribute
This error occurs if you use anything other than the letter 'L' with the *ACCESS command.

&D6 Not found
The Disk Filing System could not find the named file in the specified drive/directories.

&DC Syntax

You have typed in a command with bad arguments such as *COPY RETURN. The error message includes the correct syntax.

&FE Bad command

This means that * was omitted or that the command name was not recognized as a Disk Filing System command or utility.

12 Technical information

18-bit addressing

The BBC Disk Filing System uses 18-bit addressing giving a range from &ØØØØØ to &3FFFF. This means that two bytes and two bits are required to store a complete address, like this:

```
&3FFFF is     11   1111 1111    1111 1111    in binary
            High     Middle        Low
            bits     bits          bits
```

Each full address therefore consists of high, middle and low order bits. This is important to note, because the bits of the address are not always stored consecutively in the catalog. This is clearly shown by the way that the disk catalog is loaded into memory.

Another important factor concerns the use of a second processor. If the top two bits of an address are set, eg: &3----, the address is assumed to refer to the I/O processor.

Disk catalog

Sectors ØØ and Ø1 on the disk are reserved for the catalog. The format of the catalog is as follows:

Sector ØØ

&ØØ to &Ø7	First eight bytes of the 13-byte disk title
&Ø8 to &ØE	First file name
&ØF	Directory of first file name
&1Ø to &1E	Second file name
&1F	Directory of second file name
	. . . and so on

Repeated up to 31 files.

Sector Ø1

&ØØ to &Ø3	Last four bytes of the disk title
&Ø4	Sequence number
&Ø5	The number of catalog entries multiplied by 8
&Ø6 (bits Ø,1)	Number of sectors on disk (two high order bits of 1Ø bit number)
(bits 4,5)	!BOOT start-up option
&Ø7	Number of sectors on disk (eight low order bits of 1Ø bit number)
&Ø8	First file's load address, low order bits
&Ø9	First file's load address, middle order bits
&ØA	First file's exec address, low order bits
&ØB	First file's exec address, middle order bits
&ØC	First file's length in bytes, low order bits
&ØD	First file's length in bytes, middle order bits
&ØE (bits Ø,1)	First file's start sector, two high order bits of 1Ø bit number
(bits 2,3)	First file's load address, high order bits
(bits 4,5)	First file's length in bytes, high order bits
(bits 6,7)	First file's exec address, high order bits
&ØF	First file's start sector, eight low order bits of 1Ø bit number
	. . . and so on

Repeated for up to 31 files.

Note that the first eight bytes in each sector contain miscellaneous information about the disk. Each subsequent block of eight bytes contains information about the files, repeated for up to 31 files. The complete information about file 3 would be found in the fourth block of eight bytes on sector ØØ followed by the fourth block of eight bytes on sector Ø1.

File system initialize and !BOOT

On pressing **BREAK**, the MOS (Machine Operating System) seeks and initializes a filing system. It starts with the service ROM closest to the edge of the main printed circuit board. The first one will be initialized if just **BREAK** was pressed.

If another key is held down while you press **BREAK**, the first file system which recognizes the key will be initialized. If none recognizes the key, the CFS (cassette filing system) is initialized.

Having initialized a filing system, if 'link 5' is unmade (see below) or if **SHIFT** is not pressed and 'link 5' is made, then the start-up option of any associated device is examined. For options 1 to 3, the Disk Filing System will search for a file named `!BOOT` on the device and act according to the start-up option set.

In the case of the cassette filing system the start-up option of the ROM cartridge is examined as no start-up options are provided on cassette tapes.

Link facilities

Under the top cover of the computer to the right of the space bar, you will find eight pairs of holes in the circuit board. These are provided for the fitting of links, and give you some extra facilities, described in detail below. In the following tables, a zero means that a link is not made, and a one means that a link is made.

Links 1 and 2
Reserved for future expansion.

Links 3 and 4
The table below shows the optimum link settings for different types of disk drives which may be used with the BBC Microcomputer.

Drive type	Step	Settle	Headload	Links 3	and 4
Tandon 8Ø track	4mS	16mS	Ø	1	1
Tandon 4Ø track, Shugart	6mS	16mS	Ø	1	Ø
Teac, Mitsubishi, Olivetti 8Ø track, Siemens 211, 221, YD28Ø	6ms	16mS	56	Ø	1
Olivetti 4Ø track, Shugart SA2ØØ, BASF 61Ø6	24mS	2ØmS	64	Ø	Ø *

If you are using a drive other than those listed above, then obtain the drive speed specification from your supplier. Set the links so that each parameter is greater than or equal to the specification given. After changing the link settings, press **CTRL** **BREAK** to simulate a power down reset.

If in doubt, run at the slowest speeds (marked with an asterisk in the above list).

Link 5
See the previous section called 'File system initialize and !BOOT'.

Links 6, 7 and 8
These determine what screen mode the computer starts up in after a cold start (see the BBC Microcomputer User Guide).

Screen mode	**Links 6, 7 and 8**
Mode 7	Ø Ø Ø
Mode 6	Ø Ø 1
Mode 5	Ø 1 Ø
Mode 4	Ø 1 1
Mode 3	1 Ø Ø
Mode 2	1 Ø 1
Mode 1	1 1 Ø

13 Filing system command summary

Command	Abbreviation	
*ACCESS	(*A)	Locks or unlocks a file
*BACKUP	(*BAC.)	Copies all information from one disk to another
*BUILD	(*BU.)	Causes all subsequent keyboard entries to be stored in the named file
*CAT	(*.)	Displays a disk catalog
*COMPACT	(*COM)	Collects all files on a disk together into one contiguous sequence leaving a single block of free space
*COPY	(*COP.)	Copies all specified files from one disk to another
*DELETE	(*DE.)	Removes a single named file from the disk
*DESTROY	(*DES.)	Removes specified files from a disk in a single action
*DIR	(*DI.)	Changes the current set directory
*DRIVE	(*DR.)	Changes the current set drive
*DUMP	(*DU.)	Produces a hexadecimal listing of a file
*ENABLE	(*EN.)	Allows the use of *BACKUP and *DESTROY
*EXEC	(*E.)	Reads a disk file byte by byte as if the byte were being typed on the keyboard
*HELP	(*H.)	Displays the file system commands with syntax guidelines
*INFO	(*I.)	Displays information about specified files

*LIB	none	Selects the drive/directory for the library
*LIST	none	Displays text file on the screen with line numbers
*LOAD	(*L.)	Reads a file from disk to memory
*OPT1	(*O.1)	Switches the screen messages which accompany disk accesses on or off
*OPT4	(*O.4)	Specifies the auto-start option of a disk, relating to a file named :Ø/.$.!BOOT
*RENAME	(*RE.)	Changes a file name
*RUN	(*R.)	Runs a machine-code program
*SAVE	(*S.)	Saves a specific part of memory to the disk
*SPOOL	(*SP.)	Transfers all text subsequently displayed on the screen into a specified file
*TITLE	(*TI.)	Changes the title of a disk
*TYPE	(*TY.)	Displays a text file on screen with line numbers
*WIPE	(*W.)	Removes specified files from the disk after confirmation for each file meeting the given specification

Speech System User Guide

Introduction to the Speech System

The Speech System uses your BBC Microcomputer's sound generator, audio amplifier and speaker to 'speak' words.

The system contains a Speech Processor and a special chip called a Phrase Read Only Memory (PHROM). In this device are stored 2Ø6 ready-made words.

The Speech Processor contains a digital filter which is rather like a computer version of the human vocal tract. It relies on being fed a large number of parameters to produce the necessary sounds for making speech. Although you have direct access to the Speech Processor, it is a very highly complex task to synthesize speech, and this is why the PHROM is provided.

The actual words stored in the PHROM depend on the particular PHROM fitted. This first Speech System contains a VM61ØØ2 Word PHROM, and this is the one referred to in this Guide. Other PHROMs will become available in the future.

Take a look at Appendix A which gives a list of all the available words and word-parts stored in VM61ØØ2.

Access to the Speech System

In BASIC

This is done by using a special form of the BASIC command

```
SOUND
```

All the existing **SOUND** command facilities remain intact, but the Speech System provides a range of its own commands, which usually takes the form

```
SOUND -16,<variable>,Ø,Ø
```

In Assembly Language

This is done by using the OSWORD and OSBYTE calls in your Assembly Language programs. See chapter 4 for more details.

Using the Speech System

There are three ways in which you can use the Speech System.

You can access and manipulate the 2Ø6 words already stored in the PHROM. This can be done quickly and easily in BASIC, and is described fully in chapter 1.

You can build your own words, again by using BASIC. This requires a thorough understanding of the speech data parameters used in the Speech System. Chapter 5 explains how the Speech System works, and chapter 3 contains a program to input the speech data to the Speech System.

You can also do both the above by writing Assembly Language programs. This is explained in chapter 4, but again it is stressed that chapter 5 should be understood before attempting to build your own words.

1 Using the Speech System from BASIC

Note: From now on, any text <enclosed in parentheses> in the speech SOUND command is to tell you what sort of data is needed there. For example SOUND -16,<word number>,0,0 means type in a number in place of <word number>. Using the word number 170, the SOUND command would be SOUND -16,170,0,0.

The existing BASIC command SOUND is used to operate the Speech System, but takes a special form:

```
SOUND -16,<word number>,0,0
```

The value of this word number determines which of the 206 words is selected from the Word PHROM.

Turn to Appendix A. This contains a complete list of all the words available in the PHROM in alphabetical order. You will see that there are three columns containing: the word itself; its absolute address in the PHROM (ignore this for the time being); and the corresponding word number.

To produce the word 'Oscar', type the following:

```
SOUND -16,170,0,0
```

Appendix B lists all the words in word number order.

Now try a sentence.

To produce the letter 'I':

```
SOUND -16,41,0,0
```

To produce the word 'press':

```
SOUND -16,201,0,0
```

To produce the word 'the':

```
SOUND -16,58,0,0
```

To produce the word 'button':

```
SOUND -16,151,0,0
```

Try putting line numbers in front of each of the lines above to make a program, then run it.

```
10 SOUND -16,41,0,0
20 SOUND -16,201,0,0
30 SOUND -16,58,0,0
40 SOUND -16,151,0,0
```

Alternatively, make one line of BASIC by separating each of the five commands by a colon.

The following line of BASIC will allow you to listen to the words associated with all the ASCII codes generated from the keyboard.

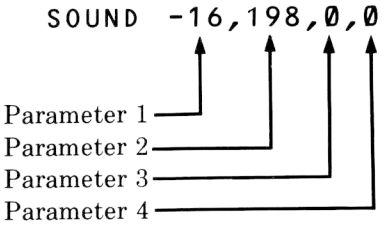
```
REPEAT : SOUND -16,GET,0,0 : UNTIL 0
```

After you have typed in this line and pressed **RETURN**, every time you press a key, you will hear the corresponding word. Remember to turn off **CAPS LOCK** and use the **SHIFT** key to get the full range.

2 The SOUND command in detail

The BASIC **SOUND** command requires four parameters. If the first parameter is a positive value, the computer treats the command as the normal **SOUND** command (see chapter 3Ø in the BBC Microcomputer User Guide). If the first parameter has a negative value, the computer treats the command as a Speech System command. Although both types of **SOUND** command use four parameters, do not confuse them – they are completely different!

The Speech System **SOUND** command parameters are expressed as in the example below, which produces the word 'mill'.



Parameters 3 and 4 are always zero, but must always be included in the Speech System **SOUND** command: the reason for their inclusion is to maintain the syntax of the **SOUND** command. However, parameters 1 and 2 can be expressed in a number of ways, and these are described below.

Parameter 1: The computer expects a four-digit hexadecimal number, and the value of this parameter tells the computer what to do with the other three parameters. There are four values you can use for parameter 1, and these are as follows:

Parameter 1 value	**Function**
&FFFx	Speak using word number in PHROM number 'x'.
&FFBx	Speak using absolute address in PHROM number 'x'.
&FF6Ø	Speak from RAM – see chapter 3.
&FFØØ	Speak from RAM – see chapter 3.

You are using Word PHROM VM61002 which has been allocated the number 0. This is equivalent to &0, so in the first two cases above, x = &0. &FFF0 is the twos complement representation of –16, so that is why –16 is used in the SOUND command. Different Word PHROMs may use different numbers, in fact any number from 0 to 15.

So far we have only used the value –16 for parameter 1. This is equivalent to &FFF0 ('speak using word number from Word PHROM VM61002'). Try replacing the –16 with &FFF0 in any of the previous examples and you will see that they are interchangeable.

Remember: Always enter the & sign before a hexadecimal number, or the computer will expect a decimal value.

The other three values of parameter 1 are explained later.

Parameter 2: Depending on parameter 1, the computer expects parameter 2 to be in one of two forms:

- Word number.
- Absolute address of the word.

Using absolute addresses

If you look at Appendix A again, you will see that beside each word there is an absolute address value. This represents the store location in the PHROM where the associated word can be accessed, and is in hexadecimal.

For example, to produce the word 'between' the command would be

```
SOUND &FFB0,&2D1B,0,0
```

Parameter 1 = &FFB0	Speak using absolute address in PHROM number &0 (decimal 0).
Parameter 2 = &2D1B	The absolute address of the word 'between' in the PHROM.
Parameters 3 and 4	Zero as always.

Note that there are two other ways to produce the word 'between':

```
SOUND -16,149,0,0
SOUND &FFF0,149,0,0
```

When you incorporate speech in your programs, you will find that the flexibility of the speech SOUND command is very useful.

3 Making your own words

Introduction

Up to now we have produced speech by using the ready-made words which are stored in the PHROM. Each word stored in the PHROM is in the form of a block of data, and this data contains all the parameters necessary for the Speech Processor to produce a word.

To make your own words, you need to do the following:

- Work out the speech parameters needed to produce your new word. To do this, you will need to understand chapter 5 very thoroughly.
- Store the above parameters, in the form of a data block, in RAM.
- Write a program to send the data block to the Speech Processor in a format it understands. The following program will do this for you.

Here is a brief explanation of what the program is doing:

PROCINIT reserves memory space called ARRAY% and assembles the assembler mnemonics. The machine code routine reverses the order of the byte held in the accumulator. Thus it would change ØØ11ØØ1Ø, for example, into Ø1ØØ11ØØ.

Lines 1Ø2Ø to 1Ø5Ø read in the speech data byte by byte, reverse the order of the bits in each byte using the machine code routine REVERSE, and store the reverse order bytes in memory starting at ARRAY%. This is essential as the operating system expects the speech data to be stored in this back-to-front way.

The rest of the procedure sends the speech data to the Speech Processor, two bytes at a time:

Line 1Ø7Ø sends the first two bytes of speech data to the Speech Processor using the SOUND &FF6Ø command. The first two bytes *must* be sent using this command as this also initializes the Speech Processor.

Lines 1Ø8Ø to 11ØØ send the rest of the speech data using the SOUND &FFØØ command. The SOUND &FF6Ø command should not be used again or the Speech Processor will be reset.

Program for sending speech data to the Speech System

PROCSPEAK(LL%)

```
1005 REM This program assumes you have
already loaded the speech data into
memory, starting at location SPEECH%.
1006 REM LL% is the length of the data
1010 DEF PROCSPEAK(LL%)
1015 PROCINIT
1020 FOR I=0 TO LL%-1
1030 A%=I?SPEECH%
1040 ARRAY%?I=USR(REVERSE) AND &FF
1050 NEXT
1060 ARRAY%?(LL%+1)=0
1070 SOUND &FF60,!ARRAY% AND &FFFF,0,0
1080 FOR I=2 TO LL% STEP 2
1090 SOUND &FF00,ARRAY%!I AND &FFFF,0,0
1100 NEXT
1110 ENDPROC
2000 DEFPROCINIT
2010 DIM ARRAY% 200
2020 DIM MC% 50
2030 FOR I=0 TO 2 STEP 2
2040 P%=MC%
2050 [OPTI
2060 .REVERSE STA &80
2070 ROL &80
2080 ROR A
2090 ROL &80
2100 ROR A
2110 ROL &80
2120 ROR A
2130 ROL &80
2140 ROR A
2150 ROL &80
2160 ROR A
2170 ROL &80
2180 ROR A
2190 ROL &80
```

```
2200  ROR A
2210  ROL &80
2220  ROR A
2230  RTS
2240  ]
2250  NEXT
2260  ENDPROC
```

Loading speech data for the word 'zero' into memory

Example program

The following example program reads the speech data for the word 'zero' into an array called SPEECH%, and uses the procedure PROCSPEAK(LL%) to send the speech data to the Speech System. PROCSPEAK(LL%) is listed on the previous page.

```
  05 REM To read the data for the word
"ZERO"
  10 DIM SPEECH% 200
  15 I=0
  20 REPEAT
  30 READ Y
  40 SPEECH%?I=Y
  50 I=I+1
  60 UNTIL Y=0
  70 REM data now in memory
  80 REM now speak the word
  90 LL%=I
 100 PROCSPEAK(LL%)
 110 END

3000  DATA 69,212,4,180,85,88,85,109
3010  DATA 129,43,17,78,53,1,241,35
3020  DATA 44,9,85,241,34,171,194,178
3030  DATA 211,104,204,49,109,34,196,89
3040  DATA 59,132,229,214,181,6,20,193
3050  DATA 57,121,174,65,150,72,46,77
3060  DATA 139,143,225,46,85,145,98,228
3070  DATA 24,108,156,58,232,185,6,21
3080  DATA 73,44,218,45,69,101,83,75
```

```
3090  DATA 190,139,17,206,109,79,21,105
3100  DATA 196,179,193,178,196,186,237,52
3110  DATA 240,46,207,19,59,81,58,149
3120  DATA 37,220,170,213,206,167,71,92
3130  DATA 235,182,52,146,209,188,246,169
3140  DATA 141,30,178,235,30,41,101,38
3150  DATA 195,204,71,74,89,73,177,16
3160  DATA 207,210,134,82,140,59,29,100
3170  DATA 221,164,160,238,169,94,47,0
```

4 Using the Speech System in Assembly Language

Introduction

There are two operating system calls which allow you to access the Speech System in Assembly Language: OSWORD and OSBYTE. For more information on the operating system calls, refer to chapter 43 in the BBC Microcomputer User Guide.

OSWORD call with A = &Ø7

Using the OSWORD call with A = &Ø7 gives you the same facilities as the BASIC command **SOUND**. As we have seen already, the **SOUND** command requires four parameters, and to use speech, parameter 1 must be one of four values special to the Speech System.

Registers X and Y are used as a register pair for sending the store locations containing the values of parameters 1 to 4 to the operating system routine, and this is done as follows:

Set up a data block which contains the least significant byte (LSB) of parameter 1, the most significant byte (MSB) of parameter 1, the LSB of parameter 2 . . . and so on up to the MSB of parameter 4.

Use registers X and Y to point to each successive store location in the data block. In the table below, the eight store locations in the data block have been called XY to XY+7.

Address	Contents
XY	Parameter 1 LSB = command type, see below
XY+1	Parameter 1 MSB = &FF
XY+2	Parameter 2 LSB = word number or address; low byte
XY+3	Parameter 2 MSB = word number or address; high byte
XY+4	Parameter 3 LSB = &ØØ
XY+5	Parameter 3 MSB = &ØØ

XY+6	Parameter 4 LSB = &ØØ
XY+7	Parameter 4 MSB = &ØØ

Note: See chapter 2.

Location XY contains the command type as for the **SOUND** command, ie:

&Fx	To speak using pointers in PHROM number x.
&Bx	To speak using absolute addresses in PHROM number x.
&6Ø and &ØØ	To speak from RAM.

The following example program causes the computer to say the alphabet (not in order).

```
 10  DIM  Q%  100
 20  DIM  H%  8
 30  FOR  Z%=0  TO  2  STEP  2
 40  P%=Q%
 50  [OPTZ%
 60  .START  LDA  #&F0
 70  STA  H%
 75  LDA  #&FF
 80  STA  H%+1
 90  LDA  #&00
100  LDX  #&00
110  .LOOP  STA  H%+3,X
120  INX
130  CPX  #&05
140  BNE  LOOP
150  LDA  #&14
160  .LOOP1  LDX  #(H%  MOD  256)
170  LDY  #(H%  DIV  256)
180  STA  H%+2
190  LDA  #&07
200  JSR  &FFF1
210  LDA  H%+2
220  CLC
230  ADC  #&01
240  CMP  #46
250  BNE  LOOP1
```

```
260  RTS
270  ] : NEXT
280  CALL START
290  END
```

OSBYTE calls &9E and &9F

The above section dealt with an OSWORD call which has the same effect as the BASIC **SOUND** command for the Speech System. It is possible to use the Speech System at an even lower level by directly writing to and reading from the Speech Processor. Before doing this you are advised to read chapter 5 very carefully. It is quite a complicated matter to control the Speech Processor, and most of the time it is best to let the operating system do so by using the OSWORD call or the **SOUND** command. However, two OSBYTE calls (call address &FFF4) have been provided to enable you to read and write to the Speech Processor.

OSBYTE call with A = &9E – read from the Speech Processor

This call allows you to read from the Speech Processor. The byte read is either from the status register or the data register of the Speech Processor, depending on the previous command sent to the Processor. To read the Speech Processor's data register you should send a 'read byte' command before making this call. Please refer to chapter 5 for a description of the 'read byte' command. If the 'read byte' command has not been sent, the status register will be read. The result is returned in Y.

OSBYTE call with A = &9F – write to the Speech Processor

This call allows you to send commands/data to the Speech Processor. The commands are discussed in chapter 5. The command to be sent should be put in Y when the call is made.

The following example procedure illustrates the use of these calls to speak from RAM (you will need to refer to chapter 3 to understand how it works). The speech data should be in memory starting at location SPEECH%. This procedure could replace the one given in chapter 3. If you have typed in that program, you can try this procedure by deleting lines 1010 to 1110 and typing in lines 1000 to 1240 below.

```
1000  DEF PROCSPEAK(LL%)
1010  PROCINIT
1020  FOR I=0 TO LL%-1
```

```
1030  A%=SPEECH%?I
1040  ARRAY%?I=USR(REVERSE) AND &FF
1050  NEXT
1060  ARRAY%?(LL%+1)=0
1070  DIM CODE% 100
1080  FOR N=0 TO 2 STEP 2
1090  P%=CODE%
1100  [OPTN
1110  .START LDY #&60
1120  LDA #&9F
1130  JSR &FFF4
1140  LDX #&00
1150  .LOOP LDY ARRAY%,X
1160  JSR &FFF4
1170  INX
1180  CPY #&00
1190  BNE LOOP
1200  RTS
1210  ]
1220  NEXT
1230  CALL START
1240  ENDPROC
```

PROCINIT should be the same as in chapter 3.

Lines 1020 to 1050 reverse the order of the bits in the speech data bytes and store the reversed bytes at ARRAY%.

Lines 1110 to 1130 send the byte &60 to the Speech Processor which is the 'speak external' command (see chapter 5). OSBYTE call &9F is used to do this.

The loop, lines 1150 to 1190, sends the reversed speech data bytes to the Speech Processor using OSBYTE call &9F.

5 Technical information

Introduction

The Speech System consists of a Speech Processor and speech data held in the associated Word Phrase Read Only Memory (Word PHROM). The words which are available depend upon the contents of the Word PHROM fitted. The first Word PHROM fitted (VM61ØØ2) contains 2Ø6 words. The complete contents of VM61ØØ2 are given in Appendix A.

The Speech Processor

The Speech Processor used in the BBC Microcomputer Speech System is made by Texas Instruments, and you are referred to the data manual available from the manufacturers for full details of its use (see Appendix D). An introduction to the Speech Processor is given here.

To create speech, the Speech Processor must be fed with suitable speech data either by the computer's CPU (central processor unit) or by direct access of the speech memory (Word PHROMs). The Speech Processor decodes the data to construct a time-varying digital filter model of the vocal tract. This model is excited with a digital representation of either glottal air impulses (voiced sounds) or the rush of air (unvoiced sounds). The output of this model is passed to an eight bit digital-to-analog converter to produce a synthetic speech waveform.

The TMS 522Ø Speech Processor may be likened to a standard microprocessor since it has a number of registers and responds to a set of commands (similar to microprocessor op-codes).

The Speech Processor registers

The TMS 522Ø Speech Processor has two input registers:

- A command register.
- A 128 bit FIFO (first-in, first-out) buffer.

And two output registers:

- A data register.
- A status register.

Command register

The command register receives a command from the CPU and holds it for the Speech Processor to interpret and execute. The Speech Processor behaves as an attached processor to the host CPU and only acts when commanded to do so. The commands are discussed later in this chapter.

FIFO buffer

The 128 bit FIFO buffer is organized as an eight bit parallel-in, serial-out buffer which can hold up to 16 bytes. This buffer is used to hold the speech data sent by the CPU when speaking under CPU control (usually from RAM).

The speech synthesizer in the Speech Processor requires the data to be in serial form starting with the least significant bit of the first byte sent. This buffer provides the required action: the bytes of speech data are sent to the Speech Processor, first byte first, and are stored in this buffer until removed from the bottom by the speech synthesizer.

A stack pointer keeps track of the last free location so that data from the CPU is always loaded in the correct place. When the stack becomes less than half full the buffer-low status bit is set (see status register). This tells the CPU that more data should be sent as the buffer will be completely empty within 25 milliseconds. This is the worst case condition corresponding to only one byte left in the buffer.

If the buffer does become empty, the buffer-empty status bit is set (see status register) and the talk status latch is reset causing speech to stop immediately. To carry on speaking under CPU control, another speak external command will have to be sent (see later in this chapter for command details).

Data register

This is an eight bit serial-in parallel-out register. It is used by the Speech Processor to form a byte of data from the serial data read from the speech data ROM (Word PHROM) during a read byte command. Data is loaded so that the last bit loaded is placed in the least significant bit of the byte.

Status register

This is a three bit register which provides information on the state of the

Speech Processor. This register may be read at any time except immediately after a read byte command. The register contents are transferred to the three most significant bits of the data bus when read (we will call these D7 to D5 inclusive). The three bits are as follows:

D7 Talk status (TS): This is high (1) if the speak command is sent, *or* if the FIFO has been loaded more than half full following a speak external command. It goes low (Ø) when the stop code (energy = 1111 – see later in this chapter) is processed, or if the FIFO buffer empties or if the Speech Processor is reset.

D6 Buffer low (BL): This is high (1) when the FIFO buffer is more than half empty. Buffer low is set when the last-in byte passes the half way mark, and is cleared when more data is loaded in so that the buffer is more than half full.

D5 Buffer empty (BE): This is high (1) when the FIFO buffer has run out of data while executing a speak external command. When this bit is set, the talk status bit is cleared and speech is terminated.

The Speech Processor commands

The Speech Processor responds to a number of commands sent by the CPU. Once the command is sent, the Speech Processor carries on and executes the command without involving the CPU. The commands available are:

Command code	Operation
xØØØxxxx	Reserved
xØØ1xxxx	Read byte
xØ1Øxxxx	Reserved
x11Øxxxx	Speak external
xØ11xxxx	Read and branch
x1ØØaaaa	Load address
x1Ø1xxxx	Speak
x111xxxx	Reset

x = don't care
a = address

These commands can be sent to the Speech Processor using the OSBYTE call with A = &9F as discussed in chapter 4.

Read byte
This command allows the CPU to access the serial data held in the speech data ROM (eg VM61ØØ2). This command causes the next eight bits to be read from the speech data ROM. The bits are placed in the eight bit data register, the last bit read being placed in the least significant bit position. This data byte is then available to be read by the CPU (using OSBYTE &9E). If another command is sent before the Speech Processor is read, the data byte will be lost and reading the processor will give the status register.

Read and branch
This command acts as an indirect load address instruction, and is issued after setting up an address (by using the load address command) whose contents are known to contain the start address of speech data. It is strongly recommended that this command should not be used as it can be guaranteed to work only when one speech data ROM is fitted.

Load address
This command allows the CPU to alter the address register of the speech data ROM (see below). The four bits labelled 'a' above are loaded into a nibble of the speech data ROM's address register. This command has to be called five times in succession to load the speech data ROM's address register fully. The least significant bit of the command (the rightmost 'a') is loaded into the least significant bit position of the nibble in the speech data ROM address register. If five of these commands are called to load the address register fully, the least significant nibble of the register is loaded first, the most significant nibble last.

Speak
This command allows speech to be generated from the data stored in the speech data ROM. The talk status bit is set and speech starts using the next available data from the ROM. The Speech Processor continues to get data and produce speech until a stop code (energy = 1111) is received. Execution of the speak command can also be stopped by the execution of a reset command.

Speak external
This command allows the CPU to supply speech data to the Speech Processor, usually from RAM. When this command is received, the FIFO buffer is flushed and data subsequently sent to the Speech Processor is put into this buffer. When the buffer is more than half full, speech begins

and continues until a stop code is encountered or the buffer becomes empty. While this command is executing all data sent to the Speech Processor is routed to the FIFO buffer and the reset command is not recognized as a command.

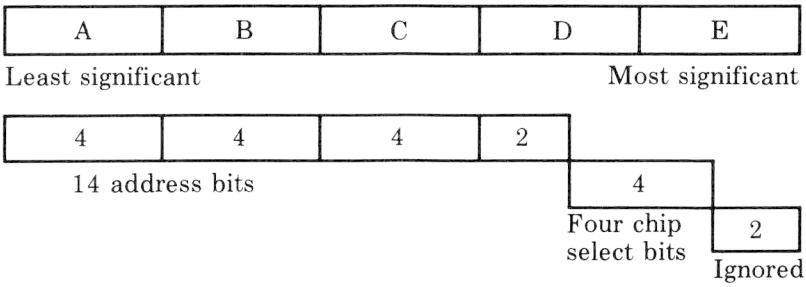
Reset
This command allows the CPU to halt a speak command and put the Speech Processor in a known state. Speech is halted immediately the command is executed. TS is cleared, BL and BE are set (high) and the FIFO buffer is purged. A load address command is sent to the speech ROM using dummy address data.

Note: The reset command cannot halt the speak external command.

Speech data ROMs

The speech data ROM used in the BBC Microcomputer is the Texas Instruments 16K VM61ØØ2, and you are referred to the data manual available from the manufacturers for full details (see Appendix D). As used in the BBC Microcomputer, this has a one bit serial output. Once the 14 bit address (of any of the 16K eight bit bytes) is written into the device, data is read out one bit at a time by toggling a control pin. After eight bits have been read, the address is internally incremented. This device is thus ideal for use with the TMS 522Ø Speech Processor which uses data in serial form.

The first thing to do before reading from the speech ROM is to load it with the address within the ROM where you want to start reading. To do this you need to send the ROM five nibbles (a nibble is four bits long). This is done by using the Speech Processor's load address command five times in succession. The least significant nibble is sent first, the most significant last. The five nibbles are used as follows:



The four chip select bits correspond to the ROM number referred to in chapter 2. VM61ØØ2 has been given ROM number Ø.

If you wanted to work at a very basic level, the following steps would have to be carried out to speak the word at, for example, address &3373 in ROM number Ø ('Oscar' in VM61ØØ2):

1. Use OSBYTE call &9F five times. Each time the call is made, Y should contain Ø1ØØØØ11 (&43), Ø1ØØØ111 (&47), Ø1ØØØØ11 (&43), Ø1ØØ11ØØ (&4C), Ø1ØØØØØØ (&4Ø) in succession. This sends the following five nibbles to the speech ROM (Ø1ØØaaaa being the load address command): 373CØ or ØØ11 Ø111 ØØ11 11ØØ ØØØØ, which as you can see sets up address &3373 and ROM number ØØØØ (Ø).

2. Use OSBYTE call &9F to send Y = Ø1Ø1ØØØØ (&5Ø) which is the speak command.

3. To terminate a word before its conclusion so as to produce sounds from parts of words, you can use OSBYTE call &9F to send Y = Ø111ØØØ Ø (&7Ø) after a suitable delay statement to reset the processor.

Speech data

The speech is divided up into a number of frames, each 1/4Øth second long. Each frame consists of a number of parameters, the maximum being one energy parameter, one pitch parameter and ten reflection coefficients (K1 to K1Ø). However, the actual values of the parameters are not stored as this would need an uneconomically large amount of memory. The parameters are firstly coded into a standard number of bits for each parameter. When the coded parameters are received by the speech synthesizer, the actual parameter value is looked up in a 'parameter look-up ROM' within the Speech Processor. The properties of the coded parameters are given below:

Parameter	Number of different values possible	Number of bits
Energy	15	4
Pitch	64	6
K1	32	5
K2	32	5
K3	16	4
K4	16	4
K5	16	4

K6	16	4
K7	16	4
K8	8	3
K9	8	3
K1Ø	8	3

Appendix C gives a table for converting actual parameter values into the coded parameters.

A number of special cases allow frames to have less than the maximum of 12 parameters. These are:

1. The repeat facility. A repeat bit follows the energy parameter in each frame. If the bit is set (1) then only the energy and pitch parameters are required, the reflection parameters of the previous frame are used.

2. Unvoiced speech (with pitch = ØØØØØØ) requires only four reflection parameters (K1 to K4). The other reflection parameters are automatically zeroed.

3. When energy = ØØØØ no other data is required. This will be the case during interword or intersyllable pauses.

With all the previous points taken into consideration we arrive at the following five different types of frame:

Frame	Energy	Repeat	Pitch	K1-K4	K5-K1Ø	Total number of bits
Voiced	eeee	Ø	pppppp	Needed	Needed	5Ø
Unvoiced	eeee	Ø	ØØØØØØ	Needed	Not used	29
Repeat	eeee	1	pppppp	Not used	Not used	11
Zero energy	ØØØØ	Not used		Not used	Not used	4
Stop code	1111	Not used		Not used	Not used	4

For example, the following would be the code for the word 'zero':

Energy	Repeat	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K1Ø
Ø1ØØ	Ø	1Ø111Ø	1Ø1ØØ	ØØØØØ	1ØØ1	Ø11Ø	1ØØØ	1Ø1Ø	1Ø1Ø	1Ø1	1ØØ	ØØ1
Ø1Ø1	Ø	1Ø11Ø1	1Ø111Ø	ØØØØØ	1ØØ1	Ø1Ø1	1ØØØ	1ØØØ	1Ø1Ø	Ø11	1ØØ	Ø11
Ø1Ø1	Ø	ØØØØØØ	11111	ØØØ1Ø	Ø1ØØ	Ø11Ø						
Ø1Ø1	1	ØØØØØØ										
1ØØ1	Ø	1Ø1Ø1Ø	11111	ØØØ1Ø	Ø1ØØ	Ø1Ø1	Ø1Ø1	Ø111	1ØØØ	Ø1Ø	1Ø1	1ØØ
1Ø11	Ø	1ØØ11Ø	11Ø1Ø	ØØ11Ø	Ø11Ø	ØØØ1	1ØØØ	1Ø11	Ø11Ø	1ØØ	1ØØ	Ø1Ø
11ØØ	Ø	1ØØØ1Ø	11ØØ1	ØØ111	Ø111	ØØØØ	1ØØ1	11ØØ	1Ø11	1Ø1	Ø11	Ø1Ø
11Ø1	Ø	1ØØØØØ	11ØØØ	Ø1Ø1Ø	Ø11Ø	ØØØØ	1ØØ1	11ØØ	1Ø11	11Ø	Ø11	Ø1Ø
111Ø	Ø	1ØØØØØ	11ØØ1	Ø11ØØ	1ØØ1	ØØØØ	Ø1Ø1	11ØØ	1ØØ1	1Ø1	1ØØ	Ø1Ø
111Ø	Ø	Ø11111	11ØØØ	Ø1ØØ1	Ø111	ØØ1Ø	1Ø1Ø	11ØØ	1ØØØ	1Ø1	1ØØ	Ø1Ø
111Ø	Ø	1ØØØØØ	11ØØØ	Ø11Ø1	1ØØ1	ØØ11	1ØØØ	Ø111	Ø1Ø1	11Ø	1ØØ	Ø1Ø
111Ø	Ø	1ØØØØØ	11ØØØ	Ø1Ø1Ø	1Ø1Ø	Ø1ØØ	1ØØ1	Ø11Ø	Ø11Ø	11Ø	1ØØ	Ø1Ø
1101	Ø	1ØØØ1Ø	1Ø11Ø	Ø1Ø1Ø	1Ø1Ø	Ø11Ø	1ØØ1	Ø111	Ø111	11Ø	1ØØ	Ø1Ø
1100	Ø	1ØØØ11	1ØØ11	1ØØ11	Ø11Ø	1Ø1Ø	Ø111	1ØØØ	1Ø1Ø	1Ø1	1Ø1	ØØ1

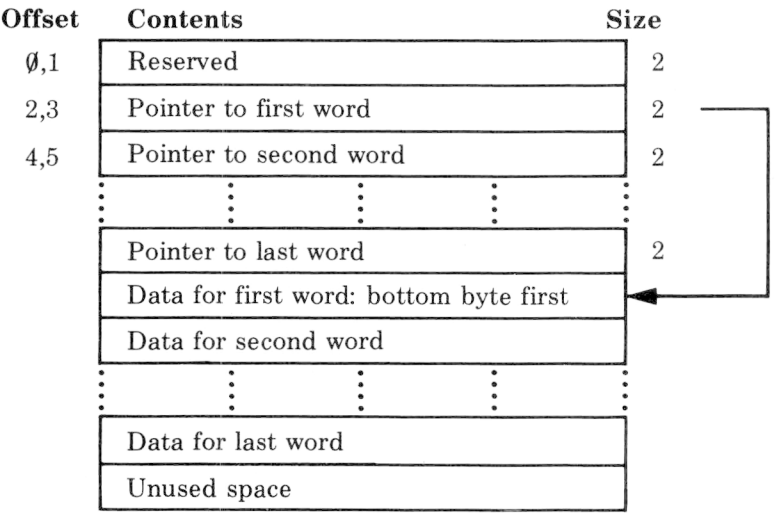
```
1100    0       100101  10011   11000   0011    0110    0101  1000  1001  011   101   011
1011    0       100110  10011   11000   0001    0111    0110  0111  1000  100   110   011
1011    0       101000  10011   10101   0010    1010    0100  1011  1011  100   101   010
1011    0       101011  10011   10101   0011    1010    0011  1010  1110  011   101   011
1011    0       110001  10100   10010   0101    1010    0011  0111  1001  111   011   010
1010    0       110001  10100   01111   0101    1001    0111  0101  1000  111   100   010
1001    0       110010  10010   01101   1000    0111    1001  1000  1000  111   010   010
1001    0       110010  10010   01101   1000    1000    1000  0110  0111  111   010   010
1000    0       110010  10010   10001   1000    0111    0110  0011  1010  110   010   011
0111    0       110100  10010   10000   0111    0111    0101  0100  1010  111   100   010
1111
```

Note: If you wanted to put this data into RAM before sending it to the Speech Processor, the bits should be formed into eight bit bytes. The first byte would be 01000101 or &45, the second 11010100 or &D4 etc. If the data does not exactly fit, the last byte should be padded out with zeros. The data is actually fed to the synthesizer in serial form via the Speech Processor's FIFO buffer and it is able to split up the bytes into the required length groups.

Word PHROM VM61002

Besides containing speech data in the form discussed in the previous section, Word PHROM VM61002 contains some other information related to the words. This includes a table of pointers, which allow the use of word numbers rather than addresses, and text strings providing yet another way to access the data. The format of the Word PHROM has been standardized as shown in the following diagram.

VM61002 Word PHROM format



Word pointers
A series of two byte pointers (low byte first), stored in order, to the word data. Note that there may be more pointers than words. This is because some words may be referred to by more than one pointer.

Word data
Speech data used by the Speech Processor to create speech. The bytes are reversed so that they are in a suitable format to be fed directly to the speech synthesizer. (Remember that data from RAM had to be reversed before being sent to the Speech Processor – the data in the ROM is already reversed so that this step is unnecessary when using the speech ROM).

Appendix A
Full list of words in Word PHROM VM61ØØ2

This Appendix lists all the words which are contained in Word PHROM VM61ØØ2 in alphabetical order.

The first column contains the word.
The second column contains the corresponding absolute address in hex.
The third column contains the corresponding word number.

Word	Absolute address (hex)	Word number
A	ØB75	39
ABORT	153C	73
ABOUT	172B	79
ADJUST	2CBA	148
ALERT	2C5F	147
ALL	216Ø	112
ALPHA	ØDØE	46
AMPS	1CA5	97
AND	3476	173
AREA	29FØ	139
AT	2ØE2	11Ø
AUTOMATIC	142E	7Ø
B	Ø6EE	2Ø
BETWEEN	2D1B	149
BRAVO	ØDE4	49
BREAK	39Ø4	187
BUTTON	2DBD	151
C	Ø899	27
CALIBRATE	39AD	189
CALL	15A6	74
CANCEL	2197	113
CAUTION	3556	176
CHANGE	1228	63

Word	Absolute address (hex)	Word number
CHARLIE	ØEBØ	51
CHECK	36Ø1	178
CIRCUIT	2A43	14Ø
CLOCK	2EØB	152
COMPLETE	1E21	1Ø2
CONNECT	2A9E	141
CONTROL	2ØØA	1Ø8
CRANE	3A21	19Ø
CYCLE	15E2	75
D	ØA1E	33
DANGER	1191	61
DAYS	21DD	114
DEGREES	3635	179
DELTA	ØD91	48
DEVICE	2E49	153
DIRECTION	3A75	191
DISPLAY	163Ø	76
DOOR	221F	115
DOWN	37CC	183
E	ØBA3	4Ø
EAST	2E97	154
ECHO	ØD4F	47
EIGHT	Ø237	3
ELECTRICIAN	2Ø71	1Ø9
ELEVEN	Ø65Ø	18
ENTER	3AE3	192
EQUAL	168B	77
EXIT	226C	116
F	Ø72Ø	21
FAIL	2ED8	155
FARAD	34CD	174
FAST	16DD	78
FEET	3B17	193
FIF-	Ø51B	14
FIRE	1D8Ø	1ØØ
FIVE	Ø342	7
FLOW	22BA	117

Word	Absolute address (hex)	Word number
FOUR	Ø1EB	2
FOXTROT	ØE41	5Ø
FREQUENCY	2F1D	156
FROM	3B56	194
G	Ø8DB	28
GAGE	22FB	118
GALLONS	26D3	13Ø
GATE	2F8B	157
GET	3B86	195
GO	1788	8Ø
GOLF	ØFØ1	52
GREEN	234A	119
H	ØA52	34
HENRY	ØF39	53
HERTZ	2914	136
HIGH	2FD4	158
HOLD	3BC8	196
HOURS	3948	188
HUNDRED	Ø425	1Ø
I	ØBD9	41
INCH	17AE	81
INDIA	1AE7	92
INSPECTOR	23B9	12Ø
INTRUDER	3ØØA	159
IS	2C28	146
J	Ø75A	22
JULIET	2737	131
K	Ø919	29
KILO	3324	169
L	ØA8D	35
LEFT	3C22	197
LIGHT	35B1	177
LIMA	ØF89	54
LINE	1371	67
LOW	17F4	82
M	ØC1B	42
MACHINE	1F7Ø	1Ø6
MANUAL	242A	121

Word	Absolute address (hex)	Word number
MEASURE	3Ø7D	16Ø
MEGA	1CEØ	98
METER	1146	6Ø
MICRO	2948	137
MIKE	1B3B	93
MILL	3C61	198
MILLI	351D	175
MINUS	1291	64
MINUTES	2D77	15Ø
MOTOR	1824	83
MOVE	24A1	122
N	Ø79D	23
NINE	Ø395	8
NORTH	3ØCE	161
NOT	12E1	65
NOVEMBER	279D	132
NUMBER	382Ø	184
O	Ø95A	3Ø
OF	3C9C	199
OFF	13C5	68
OHMS	299C	138
ON	1FCE	1Ø7
ONE	Ø2FF	6
OPEN	1869	84
OPERATOR	1A6A	91
OSCAR	3373	17Ø
OUT	388Ø	185
OVER	24FC	123
P	ØACC	36
PAPA	ØFD2	55
PASS	3121	162
PASSED	3CC5	2ØØ
PERCENT	18B2	85
PICO	1D2A	99
PLUS	2547	124
POINT	38B5	186
POSITION	3172	163
POWER	1DD3	1Ø1

Word	Absolute address (hex)	Word number
PRESS	3D11	2Ø1
PRESSURE	11E4	62
PROBE	19Ø4	86
PULL	2583	125
PUSH	31D1	164
Q	ØC5A	43
QUEBEC	1B7D	94
R	Ø7DD	24
RANGE	3D54	2Ø2
READY	194E	87
RED	2111	111
REPAIR	1E8B	1Ø3
REPEAT	25BD	126
RIGHT	32Ø1	165
ROMEO	2814	133
S	Ø988	31
SAFE	3DB2	2Ø3
SECONDS	2AEB	142
SERVICE	36CB	18Ø
SET	198E	88
SEVEN	Ø6Ø6	17
SHUT	2621	127
SIERRA	33C6	171
SIX	Ø4B3	12
SLOW	324A	166
SMOKE	14FB	72
SOUTH	3DF2	2Ø4
SPEED	19CC	89
START	1326	66
STOP	1F41	1Ø5
SWITCH	3721	181
T	ØBØ2	37
TANGO	1Ø14	56
TEEN	Ø6B8	19
TEMPERATURE	1EEC	1Ø4
TEN	Ø4E5	13
TEST	2658	128
THE	1ØDØ	58

Word	Absolute address (hex)	Word number
THIR-	Ø3ED	9
THOUSAND	Ø541	15
THREE	Ø5BA	16
TIME	13F1	69
TIMER	2BB2	144
TOOL	3292	167
TURN	3E48	2Ø5
TWELVE	Ø26A	4
TWENTY	Ø2B3	5
TWO	Ø47C	11
U	ØC8E	44
UNDER	1A16	9Ø
UNIFORM	1BC4	95
UNIT	2B55	143
UP	2BFD	145
V	Ø8Ø9	25
VALVE	376E	182
VICTOR	2874	134
VOLTS	2696	129
W	Ø9B8	32
WAIT	14AC	71
WATTS	1ØF7	59
WEST	32D1	168
WHISKEY	341D	172
X	ØB3C	38
X-RAY	1Ø6C	57
Y	ØCCC	45
YANKEE	1C42	96
YELLOW	3E8B	2Ø6
Z	Ø85D	26
ZERO	Ø19E	1
ZULU	28CB	135

Appendix B
Full list of words in Word PHROM VM61ØØ2

This Appendix lists all the words which are contained in Word PHROM VM61ØØ2 in word number order.

The first column contains the word number.
The second column contains the corresponding absolute address in hex.
The third column contains the corresponding word.

Word number	Absolute address (hex)	Word
1	Ø19E	ZERO
2	Ø1EB	FOUR
3	Ø237	EIGHT
4	Ø26A	TWELVE
5	Ø2B3	TWENTY
6	Ø2FF	ONE
7	Ø342	FIVE
8	Ø395	NINE
9	Ø3ED	THIR-
1Ø	Ø425	HUNDRED
11	Ø47C	TWO
12	Ø4B3	SIX
13	Ø4E5	TEN
14	Ø51B	FIF-
15	Ø541	THOUSAND
16	Ø5BA	THREE
17	Ø6Ø6	SEVEN
18	Ø65Ø	ELEVEN
19	Ø6B8	TEEN
2Ø	Ø6EE	B
21	Ø72Ø	F
22	Ø75A	J
23	Ø79D	N

Word number	Absolute address (hex)	Word
24	Ø7DD	R
25	Ø8Ø9	V
26	Ø85D	Z
27	Ø899	C
28	Ø8DB	G
29	Ø919	K
3Ø	Ø95A	O
31	Ø988	S
32	Ø9B8	W
33	ØA1E	D
34	ØA52	H
35	ØA8D	L
36	ØACC	P
37	ØBØ2	T
38	ØB3C	X
39	ØB75	A
4Ø	ØBA3	E
41	ØBD9	I
42	ØC1B	M
43	ØC5A	Q
44	ØC8E	U
45	ØCCC	Y
46	ØDØE	ALPHA
47	ØD4F	ECHO
48	ØD91	DELTA
49	ØDE4	BRAVO
5Ø	ØE41	FOXTROT
51	ØEBØ	CHARLIE
52	ØFØ1	GOLF
53	ØF39	HENRY
54	ØF89	LIMA
55	ØFD2	PAPA
56	1Ø14	TANGO
57	1Ø6C	X-RAY
58	1ØDØ	THE
59	1ØF7	WATTS
6Ø	1146	METER

Word number	Absolute address (hex)	Word
61	1191	DANGER
62	11E4	PRESSURE
63	1228	CHANGE
64	1291	MINUS
65	12E1	NOT
66	1326	START
67	1371	LINE
68	13C5	OFF
69	13F1	TIME
7Ø	142E	AUTOMATIC
71	14AC	WAIT
72	14FB	SMOKE
73	153C	ABORT
74	15A6	CALL
75	15E2	CYCLE
76	163Ø	DISPLAY
77	168B	EQUAL
78	16DD	FAST
79	172B	ABOUT
8Ø	1788	GO
81	17AE	INCH
82	17F4	LOW
83	1824	MOTOR
84	1869	OPEN
85	18B2	PERCENT
86	19Ø4	PROBE
87	194E	READY
88	198E	SET
89	19CC	SPEED
9Ø	1A16	UNDER
91	1A6A	OPERATOR
92	1AE7	INDIA
93	1B3B	MIKE
94	1B7D	QUEBEC
95	1BC4	UNIFORM
96	1C42	YANKEE
97	1CA5	AMPS
98	1CEØ	MEGA

Word number	Absolute address (hex)	Word
99	1D2A	PICO
1ØØ	1D8Ø	FIRE
1Ø1	1DD3	POWER
1Ø2	1E21	COMPLETE
1Ø3	1E8B	REPAIR
1Ø4	1EEC	TEMPERATURE
1Ø5	1F41	STOP
1Ø6	1F7Ø	MACHINE
1Ø7	1FCE	ON
1Ø8	2ØØA	CONTROL
1Ø9	2Ø71	ELECTRICIAN
11Ø	2ØE2	AT
111	2111	RED
112	216Ø	ALL
113	2197	CANCEL
114	21DD	DAYS
115	221F	DOOR
116	226C	EXIT
117	22BA	FLOW
118	22FB	GAGE
119	234A	GREEN
12Ø	23B9	INSPECTOR
121	242A	MANUAL
122	24A1	MOVE
123	24FC	OVER
124	2547	PLUS
125	2583	PULL
126	25BD	REPEAT
127	2621	SHUT
128	2658	TEST
129	2696	VOLTS
13Ø	26D3	GALLONS
131	2737	JULIET
132	279D	NOVEMBER
133	2814	ROMEO
134	2874	VICTOR
135	28CB	ZULU
136	2914	HERTZ

Word number	Absolute address (hex)	Word
137	2948	MICRO
138	299C	OHMS
139	29FØ	AREA
14Ø	2A43	CIRCUIT
141	2A9E	CONNECT
142	2AEB	SECONDS
143	2B55	UNI
144	2BB2	TIMER
145	2BFD	UP
146	2C28	IS
147	2C5F	ALERT
148	2CBA	ADJUST
149	2D1B	BETWEEN
15Ø	2D77	MINUTES
151	2DBD	BUTTON
152	2EØB	LOCK
153	2E49	DEVICE
154	2E97	EAST
155	2ED8	FAIL
156	2F1D	FREQUENCY
157	2F8B	GATE
158	2FD4	HIGH
159	3ØØA	INTRUDER
16Ø	3Ø7D	MEASURE
161	3ØCE	NORTH
162	3121	PASS
163	3172	POSITION
164	31D1	PUSH
165	32Ø1	RIGHT
166	324A	SLOW
167	3292	TOOL
168	32D1	WEST
169	3324	KILO
17Ø	3373	OSCAR
171	33C6	SIERRA
172	341D	WHISKEY
173	3476	AND
174	34CD	FARAD

Word number	Absolute address (hex)	Word
175	351D	MILLI
176	3556	CAUTION
177	35B1	LIGHT
178	36Ø1	CHECK
179	3635	DEGREES
18Ø	36CB	SERVICE
181	3721	SWITCH
182	376E	VALVE
183	37CC	DOWN
184	382Ø	NUMBER
185	388Ø	OUT
186	38B5	POINT
187	39Ø4	BREAK
188	3948	HOURS
189	39AD	CALIBRATE
19Ø	3A21	CRANE
191	3A75	DIRECTION
192	3AE5	ENTER
193	3B17	FEET
194	3B56	FROM
195	3B86	GET
196	3BC8	HOLD
197	3C22	LEFT
198	3C61	MILL
199	3C9C	OF
2ØØ	3CC5	PASSED
2Ø1	3D11	PRESS
2Ø2	3D54	RANGE
2Ø3	3DB2	SAFE
2Ø4	3DF2	SOUTH
2Ø5	3E48	TURN
2Ø6	3E8B	YELLOW

Appendix C
Speech parameter data coding

The entries in the bulk of the table below are the actual parameter values, the entries under 'coded value' are the values recognized by the Speech Processor and stored in speech ROMs.

Each entry for K1 to K1Ø is the real value multiplied by 512. For example, the first value under K1 is –5Ø1. This represents – 5Ø1/512 = –Ø.9785.

The pitch is expressed as a number of samples. For example, coded pitch 44 decodes to a value of 78. This means that a pitch excitation pulse is provided every 78 sample periods. At 8KHz, the sample period is 1/(8KHz) or 125 microseconds. The pitch period is thus 78 × 125 = 9.75 milliseconds, which corresponds to a frequency of 1Ø2Hz.

Energy	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K1Ø	Coded value
Ø	Ø	–5Ø1	–328	–441	–328	–328	–256	–3Ø8	–256	–256	–2Ø5	Ø
52	15	–498	–3Ø3	–387	–273	–282	–212	–26Ø	–161	–176	–132	1
87	16	–497	–274	–333	–217	–235	–168	–212	–66	–96	–59	2
123	17	–495	–244	–279	–161	–189	–123	–164	29	–15	14	3
174	18	–493	–211	–225	–1Ø6	–142	–79	–117	124	65	87	4
246	19	–491	–175	–171	–5Ø	–96	–35	–69	219	146	16Ø	5
348	2Ø	–488	–138	–117	5	–5Ø	1Ø	–21	314	226	234	6
491	21	–482	–99	–63	61	–3	54	27	4Ø9	3Ø7	3Ø7	7
694	22	–478	–59	–9	116	43	98	75				8
981	23	–474	–18	45	172	9Ø	143	122				9
1385	24	–469	24	98	228	136	187	17Ø				1Ø
1957	25	–464	64	152	283	182	232	218				11
2764	26	–459	1Ø5	2Ø6	339	229	276	266				12
3Ø4	27	–452	143	26Ø	394	275	32Ø	314				13
5514	28	–445	18Ø	314	45Ø	322	365	361				14
7789	29	–437	215	368	5Ø6	368	4Ø9	4Ø9				15
	3Ø	–412	248									16
	31	–38Ø	278									17
	32	–339	3Ø6									18
	33	–288	331									19
	34	–227	354									2Ø
	35	–158	374									21
	36	–81	392									22
	37	–1	4Ø8									23
	38	8Ø	422									24
	39	157	435									25
	4Ø	226	445									26
	41	287	455									27
	42	337	463									28
	44	379	47Ø									29
	46	411	476									3Ø
	48	436	5Ø6									31

Energy	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	Coded value
	50											32
	52											33
	53											34
	56											35
	58											36
	60											37
	62											38
	65											39
	68											40
	70											41
	72											42
	76											43
	78											44
	80											45
	84											46
	86											47
	91											48
	94											49
	98											50
	101											51
	105											52
	109											53
	114											54
	118											55
	122											56
	127											57
	132											58
	137											59
	142											60
	148											61
	153											62
	159											63

Appendix D
References

TMS 61ØØ Voice Synthesis Memory Data Manual

TMS 522Ø Voice Synthesis Processor Data Manual

Both these manuals are available from the Texas Instruments network of sales offices, or write to:

Texas Instruments Inc
PO Box 225Ø12
Dallas
Texas

Into VIEW

1 What is word processing?

The easiest way to describe word processing is to contrast it with normal typing. Think about what happens when you type a document.

First you type it in rough so that it can be edited. You edit it, changing words, swapping paragraphs around, changing paragraph lengths, adding headings, redrafting some of it and putting other sections in tabular form.

After that you type it all again. You check it, correct it, retype parts in a narrower column with side headings, perhaps retype whole pages where there are too many corrections.

After all that, you have your 'top' copy. If you want other copies, you have to photocopy, use carbons, or type it all again.

All this may seem quite normal – until you try word processing.

With a word processor, you type the text in as before, with the difference that the text appears on a monitor or television screen instead of on paper. The continual rapping and buzzing of conventional typewriters is replaced by the soft rattle of keys, and when you make a mistake, instead of going to a great deal of trouble to make a fairly adequate correction, you have only to replace one character image on the screen with another and the job is done, quickly and perfectly.

You record your rough draft on a magnetic disk or on tape, and you can cause the printer to type out copies on paper for checking.

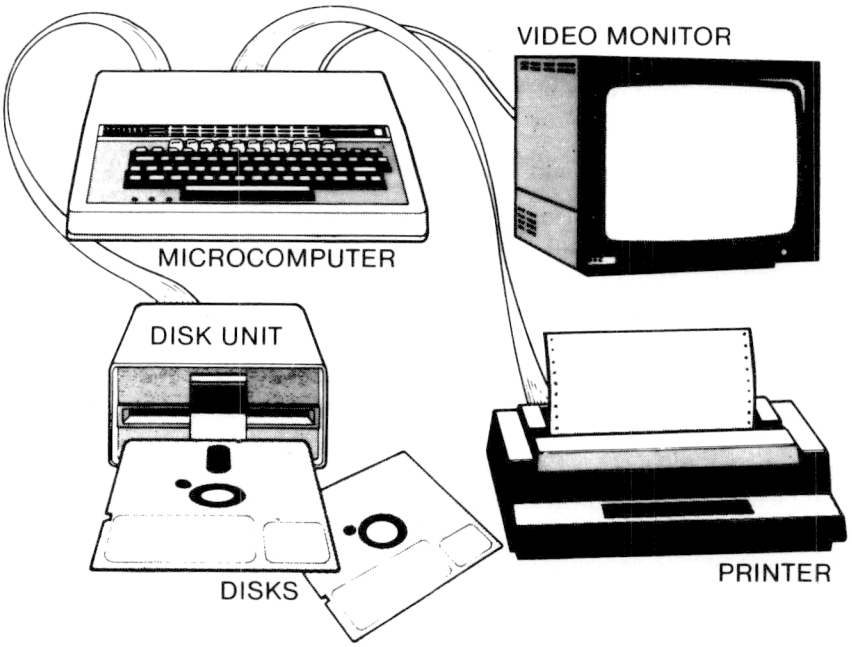
You edit the draft. When you come to 'retype' it you may find that large chunks of it are correct. With word processing there is no need ever to type these again, since they are recorded for you to use as many times as you like, in this or any other document.

You make your corrections on the screen, very easily. You insert and delete lines, move blocks of text around, and restructure the TABs even though the text is already typed! If you want to see what the text looks like in a narrower column, you can try it out in a matter of seconds.

Then you print it out again, perhaps with three 'top' copies – since it is all recorded, this is just as easy as putting another piece of paper into the machine. And if you want to send another, almost identical copy to XYZ

Inc instead of ABC Inc, with changed names throughout, there are some very cunning and very easy ways of doing that too.

The item that makes all the difference in a word processor is the microcomputer that controls it all. A typical word processor layout is like this:



All the processing is done in the microcomputer and the results are displayed on the monitor, before being recorded for future use on disks, and printed out.

2 The VIEW word processor

The VIEW word processor package is designed to do all the things described in the last chapter, and much more besides. It is especially convenient in that the processor is installed in the BBC Microcomputer. When you want word processing you have only to request it. You can even arrange your computer in such a way that when you switch on you are immediately in word processing mode.

A typical VIEW word processing layout consists of:

- A BBC Microcomputer Model B, with a VIEW word processing ROM fitted (ROM means 'Read Only Memory' and the VIEW ROM contains all the memory that is needed to convert the BBC Microcomputer into a word processor.)
- A BBC Microcomputer Disk Drive.
- A good quality black and white video monitor (for example a Kaga Denshi). Some people claim that a green screen tires the eyes less. You can also use a color monitor or, less satisfactorily, a television set.
- A printer. What printer you choose depends on your requirements. If you want the text to look as if it is typed on an electric typewriter, you should choose a good daisy-wheel printer.

To connect up your computer, monitor, disk unit and printer, see the relevant User Guides.

Before starting to use VIEW you should slip the prompt card under the clear plastic strip at the top of the computer keyboard, lining it up so that FORMAT BLOCK comes immediately above the red function key **f0**.

The prompt card will be your guide when issuing commands to VIEW while you are processing text.

BASIC

Having the VIEW word processor ROM installed in your computer does not prevent you from using the machine as a normal computer whenever you wish. To switch back to BASIC, type `*BASIC` and press **RETURN**. If you are already in BASIC and want to begin word processing, type `*WORD` and press **RETURN**.

3 Word processing with VIEW

Remember that the objective of word processing is to process words – with keyboard, screen and printer. So we shall assume that you are sitting at your computer trying everything out as you go. The more often this is the case the quicker you will learn.

Switching on

VIEW is encoded in a ROM which is plugged into one of four sockets in your computer. To get into VIEW from BASIC type `*WORD` and press **RETURN**.

Feel free to experiment with any keys – you cannot do the computer any harm by pressing them.

The command mode display

```
VIEW A2.1
No text
Editing No File
Screen Mode 7
Printer default
=>
```

The top line of the display shows the version of VIEW you are using. The words `No text` mean just that: when you enter text into VIEW or load a file from disk or tape, the words `No text` will change to show the amount of memory available. `Editing No File` will also change when you load a text file, to show the name of the file. `Screen Mode 7` is the BBC Microcomputer's Teletext mode, giving 20 lines of 34 characters each on the screen. `Printer default` refers to the printer driver. Briefly this is a program which manages printing, and the word 'default' implies that VIEW's built-in printer driver program is active. Other drivers can be loaded for particular printers. The sign `=>` shows where commands are typed in.

Text mode

If you wish to write text, you need to open up text mode. This is done by typing the word `NEW` and pressing **RETURN**.

Once you have done this, you can switch at will between command mode (the mode in which you issue general commands to the system) and text mode (the mode in which you write text). To switch between these modes press **ESCAPE**.

Notice that the words `No text` have changed to `Bytes free...` As a rough guide, one byte corresponds to one character, ie one letter, number, sign or space.

Switch to text mode by pressing **ESCAPE** and try some typing. You will soon discover that you are typing in capitals all the time. This is because at start-up the 'caps lock' is on. To switch it off press the **CAPS LOCK** key.

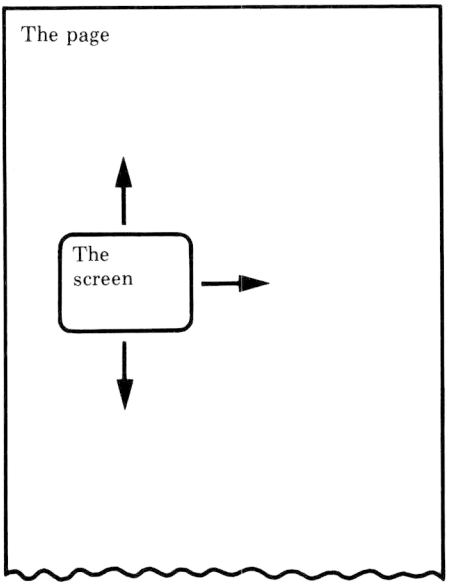
The text mode display

```
FJ .......*.......*.......*.......*.<
_
**************************************
```

The top line is the ruler, and this controls the width of the column of text under it. The asterisks in the ruler are TAB stops. The letter `F` means format and `J` means justify; these are explained in detail later in this chapter. The horizontal bar is the cursor, and shows where any new text will be typed. The row of asterisks marks the bottom of the column of text.

Screen and page

The four arrow keys move the cursor about the screen, but if you use them at this stage it is important to realize what they are doing. VIEW's text area is not limited to the screen itself. The best way is to regard the text area as a very large 'page', only a little of which is visible to you through the screen. The screen in fact is rather like a window which you can move (using the arrow keys) to any part of the page you wish.



Typing text

When you are in text mode, you can use the commands on the red function keys. Look at the prompt card, which you should keep inserted under the clear plastic clip at the top of the keyboard. The most frequently needed commands are on the bottom row, and these are used without pressing either **SHIFT** or **CTRL**. Type in anything you like and try them out.

Now let us try out some of the features of VIEW word processing. Type in the paragraph below, without ever pressing **RETURN**, since the system will move you on to another line automatically, as soon as you type a word beyond the right margin.

```
When typing in text on the VIEW
word processor, there is normally
no need to press RETURN unless you
wish to go on to another line
without having reached the margin
on the current one. Text is
automatically formatted as you
type it in. In Mode 7 it is
```

```
formatted on to a 34-character
line.
```

Notice how each line you type is justified, ie the right-hand margin is even.

(If this is not happening you have switched off the justification by mistake. At this stage the best thing to do is to return to the start-up position:

- Press **BREAK** to reset the system.
- Press **CAPS LOCK** to switch that off.
- Type NEW and press **RETURN**.
- Press **ESCAPE** to get back into text mode.

. . . and start again.)

Having typed in your text, correct any mistakes. If you made none just rewrite a bit instead. (Do not worry at this stage if any characters go beyond the right margin.)

To *replace* characters, use the arrow keys to place the cursor under the character you want replaced. Then simply type over the character and the new character will replace the old one.

To *delete* a character, place the cursor under the character you want to delete and press DELETE CHARACTER – which is function key **f9**.

To *insert* a character, place the cursor in the character space to the right of where you want the new character to be, like this:

```
WORD PROESSING
```

Then press INSERT CHARACTER (function key **f8**). The text will open up like this:

```
WORD PRO_ESSING
```

and you can insert the new character.

You can also try inserting and deleting lines. The INSERT LINE command (function key **f6**) inserts a blank line above the line which the cursor is on, and the DELETE LINE command (function key **f7**) deletes the line which the cursor is on.

TOP OF TEXT and BOTTOM OF TEXT are easily demonstrated too, as are BEGINNING OF LINE and END OF LINE. At first you will probably find yourself using the arrow keys most of the time to move the cursor about, but as you become more proficient you will find these more powerful commands save a lot of time. All these commands are described later in more detail.

When you have finished making all your changes, you will probably find that your text no longer has that neat, justified right edge you started with. Probably it looks rather like our version below.

```
When typing in a paragraph on the VIEW
word processor, there is
no need to press RETURN unless
you wish to go on to another line
without having reached the margin
on the current one. Text is
automatically formatted as you
type it in. If you are using Mode 7 it is
formatted on to a 34-character
line.
```

You can tidy up a column of text like this by reformatting it. Place the cursor anywhere on the top line and press function key f0, immediately below the words FORMAT BLOCK on the prompt card. Make sure you do not have 'shift lock' on or you will get a bleep and no action.

The whole paragraph is then changed into something like this:

```
When typing in a paragraph on the
VIEW word processor, there is no
need to press RETURN unless you
wish to go on to another line
without having reached the margin
on the current one. Text is
automatically formatted as you
type it in. If you are using Mode
7 it is formatted on to a
34-character line.
```

Of course not everyone likes justified text, and for some purposes it looks too formal. To switch off justification, hold down **CTRL** and press function key **f3** (labelled JUSTIFY MODE on the prompt card). The small letter J at top left on the screen disappears, although the F remains – we are still formatting.

To rearrange the text in unjustified lines, all you have to do is to carry out the same procedure as before: place the cursor on the top line and press the function key for FORMAT BLOCK. The result should be something like this:

```
When typing in a paragraph on the
VIEW word processor, there is no
need to press  RETURN unless you
wish to go on to  another line
without having reached  the margin
on the current one.  Text  is
automatically formatted as  you
type it in.  If you are using Mode
7 it is formatted on to a
34-character line.
```

4 Screen modes and ruler

So far everything in this book can be done in screen mode 7, which is probably the mode in which your BBC Microcomputer will start up (ie unless you have arranged for it to start otherwise).

There are in fact eight possible screen modes, and VIEW will work in any of them. In practice most people use MODE 3, since its 74-character line is very suitable for laying out typewritten material.

	Characters	Lines
MODE Ø	74	25
MODE 1	34	25
MODE 2	16	25
MODE 3	74	22
MODE 4	34	25
MODE 5	16	25
MODE 6	34	22
MODE 7	34	2Ø

To switch to another mode, type `MODE` followed by the mode number and **RETURN**.

You should be aware of the computer's memory if you are to use VIEW effectively, and the mode you are in affects the amount of memory available for text. Switch to command mode by pressing **ESCAPE** and you will see the amount of free memory shown in the form `Bytes free ...`

This shows the amount of memory left for you to use after the text already there is taken into account. A byte is a unit of memory, equivalent to a single character on the screen. However the different screen modes themselves take up different amounts of memory.

If you start in MODE 7, you have 25ØØØ bytes of memory to play with, whereas MODE 3 only allows you about 1ØØØØ and in MODE Ø you are down to about 6ØØØ before you start.

On balance MODE 3 probably offers the most useful compromise. As a rough guide, a typical A4 page takes about 2ØØØ bytes.

The ruler

For some time now you will have been aware of the ruler, and of its function in limiting the length of the line when you are typing in or formatting text.

```
.. .......*.......*.......*.......*.......*.......*.......*.......*.......*.<
```

The ruler you see when you first switch on is the default ruler, ie it is the ruler the system gives you unless you request something different. In fact you may insert a ruler into your text whenever you wish, up to a maximum of 128 rulers in any one document, and you may make them all up differently.

As we mentioned in chapter 3, the dots in the ruler are there mainly to remind you that the line is a ruler; the asterisks are TAB stops, and the angle parenthesis on the right is the margin stop.

Try calling up a default ruler for yourself, by holding down **CTRL** and pressing function key **f5** for DEFAULT RULER.

Modifying the default ruler can be done in much the same way as modifying text. You place the cursor on it by means of the arrow keys, and use such commands as INSERT CHARACTER (to lengthen the ruler) or DELETE CHARACTER (to shorten it).

Try the following to show how this works.

Get into MODE 3 by returning to command mode and typing `MODE 3` followed by **RETURN**.

Switch to text mode and call up a default ruler (function key **f5** with **CTRL**).

Now type the following paragraph.

```
Formatting relates the  length  of the text lines to the current  ruler.
Press FORMAT MODE to turn on (F is displayed at top left). Press  FORMAT
MODE again to turn off (nothing is displayed). In format mode, when text
is typed in, any  word  which overflows the right margin is transferred
whole to the following line.
```

Having typed this (or anything else if you prefer) move the cursor up to your default ruler and shorten the ruler. You can do this with DELETE CHARACTER (function key **f9**) but make sure you leave the right margin intact – this is the < at the end.

Now move the cursor on to the top line of the paragraph and press FORMAT BLOCK (function key f9). Suddenly the whole paragraph is reset to a different line length, like this:

```
Formatting relates the length of
the text lines to the current ruler.
Press FORMAT MODE to turn on (F is
displayed at top left). Press
FORMAT MODE again to turn off
(nothing is displayed). In format
mode, when text is typed in, any
word which overflows the right
margin is transferred whole to the
following line.
```

This method works equally well with justified and unjustified text. Try a few more line lengths for yourself.

Whatever you do with rulers, remember the basic rule: the system obeys the last ruler above the line that the cursor is on.

Do-it-yourself rulers

Since you are allowed as many as 128 rulers in any document you can feel free to use rulers wherever it is convenient to do so. Modifying the default ruler is often the best way to start, but having modified it once you may feel it is easier to make another adjustment to your modified ruler, rather than start from scratch with another default ruler.

Copying the current ruler can be done quite simply by holding down SHIFT and pressing COPY. The copy ruler will appear on the line which the cursor is on.

Alternatively if you want something quite unlike the default ruler you may find it easiest to make up your own ruler. To do this, first place the cursor on the line where you want it; then hold down CTRL and press function key f8 (MARK AS RULER). Two dots appear in the margin and you can make up your own ruler to the right of these.

Whenever you are making up your own ruler or modifying an existing or default ruler, you should always make sure that the finished product actually looks like a ruler. For example, it would be possible to make a perfectly valid ruler consisting simply of the two dots in the left margin

and a margin stop on the right, but it would also be fatally easy to delete such a ruler by mistake, spoiling all the text under it. So fill it in with dots at least.

We also suggest that you do not alter the left margin except for a special purpose which will be mentioned later.

Further experiments with rulers

Try out what happens if you have a very long ruler. Try extending the ruler beyond the right-hand edge of the screen, using the INSERT CHARACTER key and filling up the spaces with dots.

If you then type in text, as the cursor reaches the right-hand edge of the screen, the text to the left disappears and you find yourself typing on into blackness.

What has happened is that the screen has moved to the right over the 'page', as described in the last chapter.

In fact you may construct a ruler which is up to 132 characters wide. However you should remember that the end product of word processing is printing words onto paper, so you have to make sure that your printer can handle a line of this length.

5 Tabbing

One of the commands that the VIEW word processor has in common with a typewriter is TAB. Get a default ruler onto the screen.

```
.. .......*.......*.......*.......*.......*.......*.......*.......*.<
```

TAB stops are set at every eighth character and shown as asterisks. Like any other characters in the ruler, they can be changed. There is, for example, no reason why a ruler should not be constructed with all the TABs at one end, like this:

```
.. ...*...*...*...*.....................................................b.........<
```

(The b causes a bleep, just like the bell on a typewriter, except that VIEW allows you to place it wherever you like, and it has no connection necessarily with margins.)

The two dots in the left margin are a signal that this line is a ruler, whatever other characters happen to be on the line.

Try some TABs for yourself. Call up a default ruler and reset the TABs on it as you wish, by typing in more asterisks or cancelling those that are there already with dots. Then move the cursor to the line below the ruler and use the TAB key before you type text. You will find that so far VIEW behaves in much the same way as a normal typewriter.

Tabbing after typing

Now try this. Type a piece of text several lines long. Then place the cursor at the beginning of each line and press **TAB** .

```
.. .......*.......*.......*.......*.......*.......*.......*.......*.......*.<

        TextTextTextTextText
        TextTextTextTextText
        TextTextTextTextText
        TextTextTextTextText
```

The text immediately shoots over to line up with the first asterisk on the copy ruler at the top of the screen.

Now put the cursor back at the beginning of the line again, and press DELETE CHARACTER (function key **f9**).

A surprising thing happens – surprising to those who are only used to typewriters. The whole of the tabbing is cancelled, and the text shoots right back to the cursor position.

You can produce much the same effect by typing and tabbing as before; then instead of returning the cursor to the beginning of the line, press the black **DELETE** key on the lower part of the keyboard.

TAB characters

The reason for this behavior is that the TAB is in fact a character, an invisible character – which may seem strange until you reflect that a space is also a character and invisible. So the reason the DELETE CHARACTER key deleted the whole of the TAB is that it was deleting a single character – a TAB character.

TAB characters vary in length according to the TAB settings on the current ruler, and they can be altered, even after a tabbed line has been typed, by altering the ruler.

For example, if your ruler is like this

```
.. .......*.......*.......*.......*.......*.......*.......*.<
```

and you type in

```
JONES, Abel Baker
       The art and science of word processing
               Oxford University Press
                       March 1984
```

you will have used no TAB in the first line, one in the second, two in the third, and three in the last.

If you then change the ruler above the text to

```
.. ...*...*...*.........................................<
```

and move the cursor down again, the TAB characters instantly change size and the text jumps to

```
JONES, Abel Baker
   The art and science of word processing
       Oxford University Press
           March 1984
```

because the size of the TAB characters has been changed by the resetting of the TAB stops on the ruler.

If you wish to work with TABs a good deal you should study the relevant pages which come later in this User Guide.

Text outside the ruler

Having got this far with the placing and manipulation of text, we are ready for a rather more sophisticated operation. You have probably noticed in many reports, legal documents, and leaflets that the main text often occupies the central part of the page only, with side headings and comment to left and right.

Since we need the ruler to align and format all text, how can we produce a layout like that?

The answer is to type the text first and the headings afterwards, like this.

```
.. ..............>.........................<

                   The  left   margin   is  set
                   towards the  middle  of  the
                   page   to   leave  room  for
                   headings  and  notes  at the
                   sides.   The   text  can  be
                   justified or not as required
                   and   TABs  can  be  set  as
                   usual.
```

After the text is typed, release the margins by pressing RELEASE MARGINS (**SHIFT** and function key **f2**). Then move the cursor to the positions where you want the side headings to be, using the arrow keys. Type the headings and notes in like this:

```
.. ..............>.........................<
   SIDE HEADINGS The  left   margin   is  set  NOTES AND
   TYPED         towards the  middle  of  the  REFERENCES
   HERE          page   to   leave  room  for  TYPED HERE
                 headings  and  notes  at the
                 sides.   The   text  can  be
                 justified or not as required
                 and   TABs  can  be  set  as
                 usual.
```

. . . and when you have finished the job, don't forget to restore the margins by pressing RELEASE MARGINS (**SHIFT** and function key **f2**) again.

Note: If you are using the method described here, always format your text before placing the headings and notes. If you try to format it afterwards, VIEW will assume that you want everything in the lines concerned included in the formatting, so all your headings will be collapsed into the text and you will have to start again!

Tables and formatting

Formatting text can do a great deal of damage to tables embedded in it, unless you take action to prevent the damage. This is related to the way in which formatting deals with TABs.

Suppose you have a line with a TAB in it which you format – with FORMAT BLOCK, global format, or the formatting that occurs when you reach the end of a justified line. In such a case VIEW treats the TAB as a space, and so will redistribute spaces in the line to accommodate the TAB. Naturally this would ruin any tabular layout.

The easiest way to prevent this from happening is to start each line of your table with a single TAB, or with a few spaces. This acts as a signal to VIEW and it will not disturb the table.

If you want the table to line up with the left margin the simplest method is to place a ruler above the table with no right margin stop on it. This prevents formatting and protects the table.

6 Saving and loading files

By now you must be getting impatient at seeing your disk unit or cassette recorder standing there doing nothing, to say nothing of your printer, if you have one. So this chapter is devoted to getting them working.

Disk systems

Setting up

Before you can use disks your system must be in disk mode. It is in fact set to default to disk mode (ie this is its normal state). If it is not in disk mode you must first enter command mode, and then type `*DISK` and press **RETURN**.

Formatting

Before you can record anything onto a disk you must format the disk. The Disk Filing System User Guide tells you how to do this. Note that the formatting routine can be carried out while the system is in VIEW, using command mode.

Using disks

Saving and loading files is all done in command mode. To save a file, use

```
SAVE  filename RETURN
```

If there is another file of that name already on the disk, this command wipes it out and replaces it with the new one – which can be very useful for updating texts, but very frustrating if you do it by mistake. So always keep back-up copies of the texts you cannot afford to lose. (You can't say we didn't warn you! Of course we realize that nobody makes adequate back-up copies at first, and everybody has to learn the hard way... that's life!)

Taking a file from the disk and placing it in the computer's memory is done with the command **LOAD**:

```
LOAD filename RETURN
```

or

```
L filename RETURN
```

Before the new file is loaded, any text currently in the computer's memory is wiped. If you do not wish to wipe all the current text but want to keep it and add the new text from the file to it, you should use the command **READ**:

```
READ filename RETURN
```

So whether you **READ** or **LOAD** depends on whether you want to add to what is already in the computer's memory, or whether you want to replace it with new text.

Save a few pieces of text onto the disk for yourself. Try out **LOAD** and **READ** and watch their effects. If you want to delete any files you have saved, use

```
*DELETE filename RETURN
```

but do not delete all the files, since we need some later for printing.

A useful variant on the **SAVE** command described above is to use **SAVE** by itself. Whenever you load a file the editing line in the command mode message shows the name of the file you have loaded: `Editing filename`.

If you then modify the file and want to return it to the disk under the same name, as an updated version, you do not need to name it in the **SAVE** command. Simply type

```
SAVE RETURN
```

and it will be saved under the name given in the editing line. This can be a time saver, but it is always worth checking the editing line to make sure that the name there is really the one under which you want to save the text.

Locking files

If there are any files you particularly wish to preserve, you can 'lock' them with the command

```
*ACCESS filename L RETURN
```

Try this with one of the files you have saved. Once a file is locked, you cannot update it by saving another file of the same name. If you try to, the system replies `File Locked`. To unlock it, use the `ACCESS` command again without the `L`.

To find out what files are on the disk (including which files are locked) type

```
*. RETURN  or  *CAT RETURN
```

Locked files are marked with the letter `L`.

Help!

It may be helpful to know that there is a list of disk commands available on the screen, which can serve as a rapid reminder. Put a (formatted) disk into the disk drive. Then type

```
*HELP DFS RETURN
```

However, for the finer points of disk operation you should consult the Disk Filing System User Guide.

Cassette tape

If at all possible, use a cassette recorder with a remote control (REM) socket. Otherwise you will find yourself constantly rewinding and estimating positions on the tape.

Before you can use a cassette recorder, your system must be in tape mode. You must first enter command mode and type

```
*TAPE RETURN
```

Recording files

Make sure there is a blank tape in the recorder.

Type

```
SAVE filename RETURN
```

The 'cassette motor' lamp turns on. The message appears on the screen:

```
RECORD THEN RETURN
```

Press RECORD on your cassette recorder and **RETURN** on the computer. When the prompt (=>) reappears, the file is recorded.

If your cassette recorder has no motor control (REM) socket, stop it quickly. If it has a REM connection it will stop automatically.

Reading files

When using VIEW with cassettes, there is no **LOAD** command, ie no command which automatically wipes the current text from memory and replaces it with new text from a tape file. Instead there is the **READ** command which adds new text from the file to any text currently in memory. If you wish to wipe text from memory first type **NEW** **RETURN**.

Wind back the cassette to the appropriate point.

Type

```
READ filename RETURN
```

The 'cassette motor' lamp comes on.

Press PLAY on the cassette recorder. Depending on the cassette recorder you may hear high or low pitched sounds. When the prompt (=>) reappears the file has been read in.

Printing

If you have difficulty in printing out files from cassette, in particular if you keep getting **Block?** messages, you may be able to solve the

problem by an alternative method of recording onto cassette. This is covered in the next chapter.

Names of files

One final point affecting both disk and tape: how should you name your files?

One thing to beware of is the use of spaces in file names. For example, you might type the first chapter of a report and save it under the name 'ABC'; you might then go on to type the next chapter and save it in a second file which, quite naturally, you call 'ABC 2'.

Unfortunately the computer takes the space between the C and the 2 to indicate that this is the end of the file name. To the computer, therefore, you have saved one file called 'ABC' and then another file also called 'ABC', which overwrites the first. The cure for this is to miss out the space and call the second file 'ABC2'.

Think out your file names carefully. Ideally they should be unique. If some of them relate to each other (like chapters in a report) they should reflect this fact. It is also best if they actually mean something and are not just random sequences of letters which you will have difficulty remembering.

7 Printing

Printing can be done in two ways:

- Printing out whatever text is currently held in the computer's memory.
- Printing directly from a file on disk or tape.

If you print directly from a file on disk or tape, this has no effect at all on the text in memory, so there is no reason why you should not pause while editing the text in memory to print out text from files.

For general advice on the management of printers see the manual for the printer itself. In these pages we shall concentrate on the word processing aspects.

When you have everything set up so that you can print out text quickly and easily, printing will seem a very straightforward operation, but there is a good deal more going on behind the simple command PRINT.

VIEW does not itself directly manage the printer. Instead it sends codes to a printer driver program which in turn sends codes to the printer to produce the effect you have specified in your text and stored commands. This may seem an unnecessarily complicated way of running things, but it has an important benefit: VIEW can be made to work with many different printers, simply by using different printer driver programs.

In fact VIEW itself contains a default printer driver program, which is adequate for straightforward printing on most printers. But if you are using a high quality daisy-wheel printer such as a RICOH or a QUME you need some way of taking advantage of the special effects these printers can provide – such as bold or underlined type.

This is done by installing a special printer driver program for that printer. Programs are available for many printers, and the Printer Driver booklet goes into much more detail on their use and how they work.

General procedure for printing

If you need a printer driver to run your printer, load it from disk or tape. The command is PRINTER followed by the name of the printer and RETURN. For example:

```
PRINTER RICOH RETURN
```

If your printer is of the serial RS423 type, the computer must be switched to this type of output. The printer driver may do this for you, but space for the driver's facilities is limited, so you may have to select RS423 output for yourself. See the Printer Driver booklet. To select for RS423 output type:

```
*FX5,2 RETURN
```

For similar reasons you may have to set the baud rate. This is done with a command beginning `*FX8` followed by a digit selecting the rate as follows.

Baud rate	Command
75	`*FX8,1`
150	`*FX8,2`
300	`*FX8,3`
1200	`*FX8,4`
2400	`*FX8,5`
4800	`*FX8,6`
9600	`*FX8,7`
19200	`*FX8,8`

Your printer may allow microspacing. The normal way in which VIEW justifies a line is by distributing any spaces left over throughout the line, adding the extra spaces to some of the spaces already present in the line. This is the standard practice because many printers only have whole character spaces.

Some printers however have a facility for adjusting the position of characters by 120th of an inch. This allows VIEW to adjust the line more finely by dividing the spaces left over into units of 120th of an inch and distributing these units evenly throughout the line, giving a much neater appearance.

This is microspacing. If your printer and printer driver have this facility, you can call it into action by typing `MICROSPACE` **RETURN**. If you are not sure, type it anyway, and if it is not available VIEW will tell you so.

Use the command `PRINT` or `SHEETS` to print the text, as described below.

PRINT

This is very straightforward to use. It prints out the entire file from disk or tape, or the entire text in memory.

To print out the text in memory, use `PRINT` **RETURN**.

To print a file, use `PRINT filename` **RETURN**.

When VIEW is printing and comes to the end of a page, either because the set number of lines is used up or because it encounters a 'page eject', it carries out a page eject (ie the printer winds on whatever number of lines would be needed to take care of the bottom margin on one page and the header margin on the next) and just carries on printing. Headers and footers, and other matters concerned with page layout are dealt with in chapter 9.

If you use `PRINT` with several files in succession, you can set automatic page numbering (see chapter 9) at the beginning of the first file and it will continue throughout the batch.

To stop printing, press **ESCAPE**.

SHEETS

If you are using separate pages fed into the printer one at a time you will have to use this command. It is also useful when you do not want to print all the pages in the file, since it allows you to miss out pages.

To print out the text in memory, type `SHEETS` **RETURN**.

To print a file type `SHEETS filename` **RETURN**.

The prompt appears: `Page 1..`

To print, press any key except M, Q, **ESCAPE**, **COPY** or **BREAK**.

To miss out a page, press M.

To stop printing, press Q.

When page 1 is printed or missed out, VIEW goes on to prompt `Page 2..` etc, until the file is finished.

Editing procedures

The fact that VIEW allows you to print out both files from tape or disk, and from text in memory, using either PRINT or SHEETS can be useful for revising a lengthy report, for example.

Suppose you have already printed out pages 1 to 2Ø and subsequently want to revise page 9. You load the file and carry out your revisions. You record the amended file back onto the disk.

All you need print again is page 9. The simplest way of doing this is to delete everything in memory coming before and after page 9, by setting markers 1 and 2 each side of the text to be deleted and pressing DELETE BLOCK (CTRL fØ) This is quite safe since the full version remains on the disk. Then use PRINT or SHEETS alone, and page 9, which is all that remains in memory, will be printed.

SCREEN

The SCREEN command allows you to 'flick through' the pages of a file, or of the text in memory, to get an idea of how it will be printed out. In this way you can check the formatting and see where the page breaks occur.

To display the text in memory, type

```
SCREEN RETURN
```

To display the contents of a file, type

```
SCREEN filename RETURN
```

To display the contents of several files, type

```
SCREEN filename1 filename2 filename3 RETURN
```

As soon as you press RETURN the first page moves up the screen. To move on to the next screenful press SHIFT . If you have specified headers or footers (for example, page numbers) these will be displayed.

Highlights

We mentioned earlier that some printers permit special effects such as underlined or bold type. Text to be printed in such a way is marked using the HIGHLIGHT 1 and HIGHLIGHT 2 keys.

When the text is printed, these highlight codes signal to the printer driver that a special effect is required, and the printer driver sends instructions to the printer.

To mark highlight 1:

- Place the cursor under the first character of the text concerned and press HIGHLIGHT 1. The text will jump to the right and an underline will appear.
- Move the cursor to the space after the last character of the text concerned and press HIGHLIGHT 1 again. A second underline will appear and if there is any text to the right it will move to make room for the underline.

Text marked for highlight 1 looks like this on the screen:

```
See the _VIEW Guide_ for details.
```

and would normally result in:

```
See the VIEW Guide for details.
```

To mark highlight 2 use the same procedure as for highlight 1. Instead of an underline the marker in the text is an asterisk. Text marked for highlight 2 looks like this on the screen:

```
See the *VIEW Guide* for details.
```

and would normally result in:

```
See the VIEW Guide for details.
```

Resetting highlight codes

We say that these would 'normally' be the results because some printer drivers and printers have other facilities than underlining and bold type – facilities such as an additional character set or superscripts.

Since there are only two highlight codes, VIEW allows you to reset these codes temporarily to send alternative signals to the printer driver. The stored command `HT` is used for this, and is described in more detail later.

Printing from cassette

When printing from cassette VIEW reads in one block of text at a time and prints it before reading in the next block. On some cassette recorders the BBC Microcomputer cannot stop the cassette motor quickly enough to prevent the tape running on to the next block. If it moves into the text block you will get a `Block?` error message.

Obviously one way to avoid the problem is to load the file into memory and print out from memory, as described above.

Alternatively you can re-record the file, leaving more tape between blocks. This is done as follows.

Type: `*OPT 3.10` **RETURN**

Type: `WRITE filename` **RETURN**

For more information on `*OPT` consult the BBC Microcomputer User Guide.

8 Moving and changing text

One of the pleasures of word processing is the ease with which you can modify text – inserting words, moving whole paragraphs from one point to another, copying sections and deleting them.

We have already done a little of this, inserting and deleting a character at a time, but VIEW has much more to offer.

Type in the following example to try out the more powerful commands.

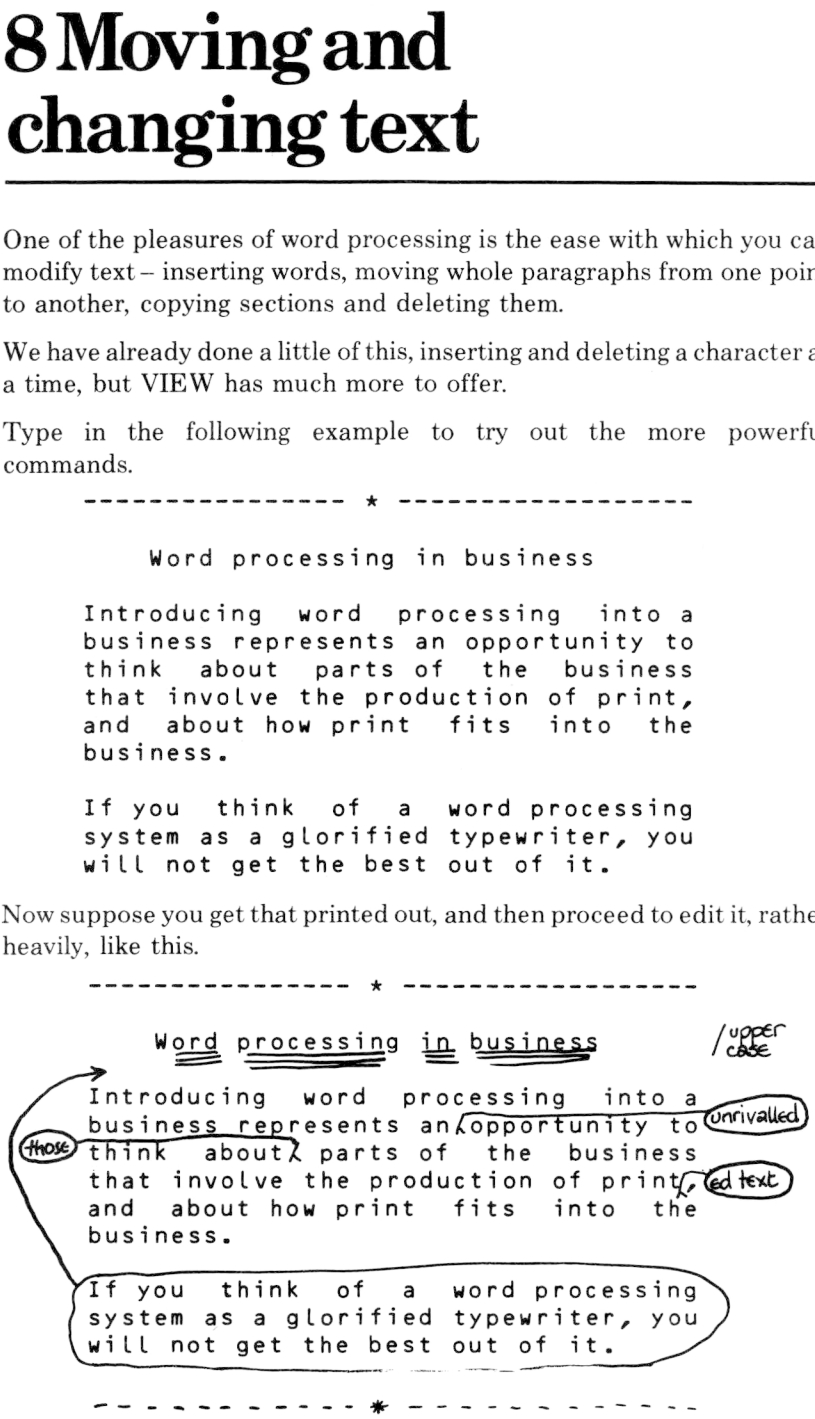
```
---------------- * ------------------

    Word processing in business

Introducing word processing into a
business represents an opportunity to
think about parts of the business
that involve the production of print,
and about how print fits into the
business.

If you think of a word processing
system as a glorified typewriter, you
will not get the best out of it.
```

Now suppose you get that printed out, and then proceed to edit it, rather heavily, like this.



Editing text like this is done by moving the cursor and using the red function keys for commands such as INSERT CHARACTER, DELETE CHARACTER, INSERT MODE, SWAP CASE, and the commands involved in moving blocks of text such as MOVE BLOCK.

Changing the title to upper case letters only is a simple matter – the SWAP CASE command is indicated. Place the cursor under the 'o' of 'Word' and press **SHIFT** **f1**. The case changes from lower to upper and the cursor moves on. All you do is to keep pressing SWAP CASE until the whole title is converted. In practice of course you might decide to retype a short heading like this, but remember that retyping is where many mistakes occur.

Now to change 'an opportunity' into 'an unrivalled opportunity'. Place the cursor under the first 'o' of 'opportunity' and press **CTRL** **f4**. This puts VIEW into INSERT MODE so that any characters typed will cause the text to the right to move right to make room for the new text. A letter `I` appears at top left on the screen to remind you. Type in the additional word with a space after it and move on to the other two corrections which can be dealt with in the same way. When you have done them all, cancel INSERT MODE by pressing **CTRL** **f4** again.

The next job is to move the three lines at the bottom to the top under the title. Operations such as moving, copying or deleting blocks of text are controlled by setting markers. To move a piece of text, set markers 1 and 2 before and after it as follows:

- Place the cursor on the line above the text you wish to move.
- Press SET MARKER (**SHIFT** **f7**) and `MK` appears at top left on the screen.
- Press 1.
- Place the cursor on the line below the text you wish to move.
- Press SET MARKER again.
- Press 2.

Carry out these operations slowly and carefully. If you make a mistake you can just set the markers again. You can also cancel markers 1 and 2 by changing to command mode and typing `CLEAR` **RETURN**.

Having set markers 1 and 2, move the cursor up to the blank line under the title and press MOVE BLOCK (**SHIFT** **f0**). The three lines will appear there and the other lines will open up to make room for them.

Finally we want to copy the rule design at the top and place it at the bottom. Once again we set markers above and below the rule, and place

the cursor where we want it copied to, but this time instead of pressing a red function key we press the black **COPY** key at bottom right on the keyboard.

After that the text should look like this.

```
---------------- * ------------------

     WORD PROCESSING IN BUSINESS

If you think of a word processing
system as a glorified typewriter, you
will not get the best out of it.

Introducing word processing into a
business represents an unrivalled opportunity to
think about those parts of the business
that involve the production of
printed text,
and about how print fits into the
business.

---------------- * ------------------
```

To justify the text again you have only to reformat the second paragraph by placing the cursor anywhere on the top line of that paragraph and pressing FORMAT BLOCK (function key **f9**) and the job is done.

```
---------------- * ------------------

     WORD PROCESSING IN BUSINESS

If you think of a word processing
system as a glorified typewriter, you
will not get the best out of it.

Introducing word processing into a
business represents an unrivalled
opportunity to think about those
parts of the business that involve
the production of printed text, and
about how print fits into the
business.

---------------- * ------------------
```

9 Stored commands and page layout

We have already used many immediate commands, such as DELETE CHARACTER and SET MARKER, and some command mode commands such as SAVE and LOAD.

VIEW also has a third set of commands which are entered in text mode, but do not have any effect at all as they are entered. These are stored commands, and their purpose is to influence the way the document is printed.

Perhaps the most obvious in its effect is PE, which means 'page eject'. This is a command to the printer to stop printing and eject the page.

Stored commands are used in text mode, and they appear in the left margin, under the two dots to the left of the ruler. It is as if they were marginal notes, telling the printer what to do as it progresses down the text.

The stored command page eject would be entered like this:

- Position the cursor on the line where you want the command.
- Press EDIT COMMAND (SHIFT f8).
- The cursor moves into the left margin.
- Type PE RETURN.

When the file is printed, as soon as the printer comes to the line on which you entered PE the page will eject.

Some of the other stored commands are used like this, some with numbers beside them, and some with text. For example suppose you want to tell the printer to work to a page length of 45 lines. The stored command for this is PL (page length) and it is entered in the margin just like PE. Beside it on the same line goes the number of lines: PL 45.

- Press EDIT COMMAND (SHIFT f8).
- The cursor moves into the left margin.
- Type PL RETURN.
- Type 45.

Similarly if you want to center the words 'The United States' on a line,

you can do so by typing it on the left and placing the stored command CE beside it, like this:

```
CE  The  United  States
```

If you want to range text to the right, as for example with instructions on a form, you would use the stored command RJ like this:

```
RJ Membership subscription $.........
RJ Handbook $.........
RJ Badge $.........
RJ Magazine $.........
RJ ---------
RJ TOTAL $.........
```

The effect when printed would be:

```
Membership subscription $.........
                Handbook $.........
                   Badge $.........
                Magazine $.........
                         ---------
                   TOTAL $.........
```

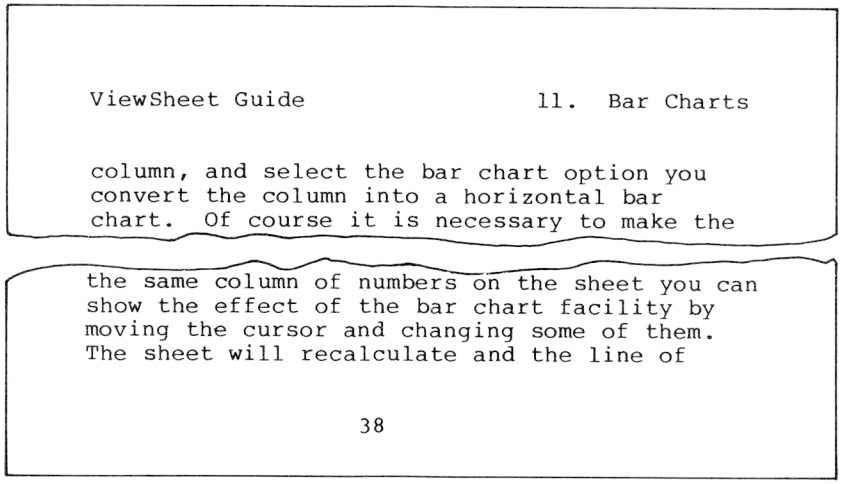
Line spacing can also be controlled with stored commands. VIEW normally assumes solid text, but if you want part of your document with extra spacing between the lines you can use the command LS followed by the number of lines spacing, eg LS 2.

Book and report work

Stored commands come into their own when you are processing books and reports, although they can be very useful for letters too. In book and report work, you need to have at your disposal at least some of the features normally associated with the printing of books, such as top and bottom margin setting, lines at the head and foot of the page (headers and footers), and methods of emphasizing type.

VIEW provides all these. In its standard (default) layout it assumes four blank lines at the top of the page, followed by a header line, containing the title or series heading for the page. Below this are another four blank lines, after which the text begins. Footers (normally containing page numbers and report codes) are similarly separated from the text and from the bottom of the page by four blank lines.

Of course all these spacings can be changed and the headers and footers themselves can be cancelled if you wish. Remember that once you define a header or footer, VIEW will continue to print it until you issue other instructions. So if you define your header as 'CHAPTER 3' you will have every page after that headed 'CHAPTER 3' until you redefine the header as 'CHAPTER 4'.



The illustration shows a page from a guide to VIEW's associated package 'ViewSheet'. To make up this page all spacings have been reset, and headers and footers have been defined.

The top margin has been reset to three spaces, instead of the default four, by the stored command `TM 3` which is used in the same way as the page length command described above.

The header margin, ie the lines between the header and the text, has also been reset with `HM 3`. Footer and bottom margins have been similarly reset to two and three lines respectively with the commands `FM 2` and `BM 3`.

The header has been defined so as to display both the title of the book and the title of the section. Headers always have three components: left, center and right.

The define header command is `DH`, and when this is used it must always be followed by the three components divided by spacers (normally the slash `/`). If one of the components is blank, simply place two slashes together. So the general rule is:

`DH /`left component`/`center component`/`right component`/`

In the present case the header would have left and right components but no center:

```
DH /ViewSheet Guide//11. Bar Charts/
```

Footers can be set in exactly the same way as headers, and also have left, center and right components.

Once a header or footer has been defined VIEW will print it on every page until it is cancelled. To cancel a header use the stored command HE; to cancel a footer use FO.

Number registers

The footer in the last example is a page number, which may seem strange: surely it would defeat the whole purpose of footers if you had to redefine the footer for every page.

In fact it is possible to set a number register P to the number of your first page, and it will automatically increase by one every time the printer completes a page. All you have to do then is to define your footer so as to print out the value of P and your page number will be printed automatically.

VIEW has 26 number registers, labelled A to Z. Only two of these are allocated: P for pages and L for lines. The rest you can set as you wish.

To set register P to value 1 for the first page, use the stored command SR (set register), like this:

```
SR  P  1
```

You have then only to define the footer as the value of P. In order to distinguish between the character P and the value of register P you must place the vertical line symbol in front of it. So the footer is defined as:

```
DF  //P|//
```

That is to say: no left component, centered page number, no right component.

Other stored commands which can cause number registers to be printed out are: DH (define header), CE (center), RJ (right justify) and LJ (left justify). In fact the command LJ exists simply for the purpose of printing out registers, since VIEW always justifies left unless told to do otherwise.

1Ø Macros

One of the main reasons for using word processors is that you can make the computer do all the really boring jobs.

Suppose you have to type the same letter or invitation 3Ø times over. The chances are that you will make half a dozen mistakes in your letters which you will only discover when it is too late.

VIEW offers a facility which avoids all such problems. It allows you to make up any collection of text and commands into a package. You give the package a name, and you can then cause the whole package to be printed as often as you like, merely by entering that name in the margin like a stored command.

Such packages are known as 'macros'. To show how they work, suppose you have to send out innumerable copies of the following invitation.

```
ABC Computers Inc
request the pleasure of your company
at the launching of the

LOGOMANIAC

word processing package
at the Queen's Hotel
on Friday 13th February at 12.00 noon
```

Obviously you would use the **CE** command heavily to center all the lines, like this:

```
CE   ABC Computers Inc
CE   request the pleasure of your company
CE   at the launching of the

CE   LOGOMANIAC

CE   word processing package
CE   at the Queen's Hotel
CE   on Friday 13th February at 12.00 noon
```

To make this into a macro so that you could print out many copies with the least possible trouble you need to:

- Give it a name.
- Signal to VIEW that it is a macro.
- 'Call' it, ie to tell VIEW to use it.

Names of macros can consist of any combination of two letters that is not already used as a stored command. The stored command to define (give a name to) a macro is **DM**, so to name the macro AZ we simply place before it the stored command: **DM AZ** and after it the stored command **EM** (which of course means 'end macro').

The finished macro therefore looks like this:

```
DM   AZ
CE   ABC Computers Inc
CE   request the pleasure of your company
CE   at the launching of the

CE   LOGOMANIAC

CE   word processing package
CE   at the Queen's Hotel
CE   on Friday 13th February at 12.00 noon
EM
```

When you have written, defined, named and ended your macro, you have only to use it. As it stands it will not print out at all. To make it print you have to call it, ie to enter its name in the margin as if it were a stored command. Press EDIT COMMAND (**SHIFT** **f8**), type **AZ** and press **RETURN**.

Try entering it several times as a test. When you set up the printer and type **PRINT**, the text will be printed out as many times as you have entered the macro's name in the margin.

Modified macros

When composing standard letters, invitations and other documents, we frequently need a piece of text which is essentially the same in each example, but has to be modified so as to 'personalize' it. Normal copying techniques fail here, but VIEW has a special facility which makes its macros even more effective.

Suppose we want to make up a standard, all-purpose invitation, which can be used to invite anybody to anything.

To start with we shall try a macro with just one modified item. Suppose we want to design an invitation into which any name can be inserted, an invitation that begins like this:

```
          ABC  Computers  Inc
request the pleasure of the company of
          Mr Bill Brewer
```

In cases like this VIEW allows you in effect to leave blanks in the macro into which you can fit any name you like at the printing stage. The method is to replace the words which are to be changed with the symbols @Ø, @1, @2 . . . @9. Each time the macro is called for printing, these parameters are filled in by typing the words on the same line as its name.

In this case we would simply place @Ø in place of the name:

```
DM   BY
CE   ABC Computers Inc
CE   request the pleasure of the company of
CE   @Ø
CE   at the launching of the

CE   LOGOMANIAC

CE   word processing package
CE   at the Queen's Hotel
CE   on Friday 13th February at 12.ØØ noon
EM
```

Later in the text, you enter the macro as if it were a stored command, with the appropriate name beside it.

```
BY   Mr Bill Brewer
BY   Ms Jan Stewer
BY   Mr Peter Gurney
```

After that, all you have to do is to issue a print command and the macro will be printed, its blanks filled in, as many times as you have entered its name in the margin.

Developing this into a genuinely all-purpose invitation is merely a matter of extending the number of parameters, so as to leave blanks for almost everything.

```
                  ....................
  requests the pleasure of the company of

                  ....................

                   on the occasion of
                  ....................
                  ....................
        at ..........................
     on ................at.............
```

Try it yourself, before looking at the finished macro below.

```
DM   CX
CE   @0
CE   requests the pleasure of the company of

CE   @1

CE   on the occasion of
CE   @2
CE   @3
CE   at @4
CE   on @5 at @6
EM
```

When you come to use it the method is much the same as before:

```
CX   XYZ Computers Inc,Jim Smith,the launching,of ASTROCALC,
```

. . . and so on.

One variation is worth noting. Suppose we want to invite Tom, Dick and Harry, who naturally go everywhere together. What about the comma after 'Tom'? Commas are used as spacers between parameters, so if we put it in it would have the wrong effect. To avoid this problem VIEW has the law that if you want to include commas in a parameter you place it in angle parentheses, like this:

```
CX   XYZ Computers Inc,<Tom, Dick and Harry>,launching ...
```

Apart from this there is virtually no restriction in the use of macros. The macro below, for example, consists almost entirely of commands. Its purpose is to provide an automatic layout for the top of a letter.

```
DM    LH
RJ    @0
      @1
      @2
      @3
      @4
RJ    @5

      Dear @6

CE    @7

EM
```

Try for yourself setting out the parameter line to make the beginning of the letter read:

```
                                        25 June 1984

John Smithson
490 Wynn Drive
Phoenix
AZ 84071
                                        Ref.25/3

Dear Mr Smithson

                    Lease Agreement
```

Macros for mail shots

Using macros in this way, you can develop quite a sophisticated system for standard letters, all of which can be individually addressed and have a number of different key words and numbers in their contents.

Suppose we wish to send the following letter to applicants for a course, making changes of date, student, course, fee, etc.

```
Mr A B Carter                    15 Sept 1983
270 Terminal Ave
Fairfax
VA 23131

Dear Mr Carter

Thank you for your application. We
confirm your enrolment for the
Intermediate Course C.  Your student
number is 552.

You will receive further details in
a few days.  Meantime will you please
send the acceptance payment as
shown in our brochure of $22.50.

If you have any enquiries please
address them to Dept 4B.

Yours sincerely
```

The equivalent macro would be:

```
DM  LL
    @0                                  @1  Sept  1983
    @2
    @3
    @4

    Dear @5

    Thank you for your application. We
    confirm your enrolment for the
    Intermediate Course @6.  Your student
    number is @7.

    You will receive further details in
    a few days.  Meantime will you please
    send the acceptance payment as
    shown in our brochure of @8.

    If you have any enquiries please
    address them to Dept @9.

    Yours sincerely

EM
```

and appropriate parameter lines would be as follows:

```
LL Mr A B Carter,15,270 Terminal Ave,Fairfax,VA 23131,Mr Carter,C,552, $22.50,4B

LL Ms Jane Brown,15,197 Bishops Way,Beaverton,OR 97116,Ms Brown,B,453, $19.25,9A
```

Automatic layout

If you need to produce reports or books to a standard format, it is possible to use macros in conjunction with number registers to automate the layout in quite a sophisticated way.

Suppose you produce a series of reports, each with many chapters with headings and sub-headings, all numbered as follows:

```
CHAPTER 1

Chapter title

1.1     Section heading

1.1.1   Sub-section heading
        Text text text text text text text text text text text text
        text text text text text text text text text text text text
        text text text text text text text text text text text text

1.1.2   Sub-section heading
        Text text text text text text text text text text text text
        text text text text text text text text text text text text
        text text text text text text text text text text text text

                                                              Page 1
```

The most efficient way to manage such a series would be to set up a file containing macros to control all the standard features: spacing, bold type, and in particular numbers. For example if we set register C to 1, and then cause a macro to increase register C by 1 each time a chapter heading is printed, we can print chapter headings completely automatically. Numbers for section and sub-section headings, and page numbers could be dealt with in the same way.

In the case of chapter headings the setting of the register would take the form:

```
SR   C 1
```

and each time a chapter heading was printed a macro would be called containing the line:

```
SR   C ¦C+1
```

which means that the new value of C is set to 1 greater than the old value of C. Later in the macro the value of C would be printed out by

```
LJ   *CHAPTER ¦C*
```

which uses the LJ (left justify) command to print out the chapter heading, the asterisks to print it in bold, and the vertical bar to indicate that it is register C that is being printed, not the letter C.

Using these principles we can construct a set of macros to manage a whole series of reports. The macros would be saved on disk, and used as part of the print command whenever a report was printed. For example if the set of macros is given the name 'BOOK' and the report files are 'A1' to 'A5' the print command would be:

```
PRINT BOOK A1 A2 A3 A4 A5
```

The macros below are an example of how this can be done. The way they do this may not be immediately apparent, but they will repay study.

Setting-up sequence:

```
DF   ///Page¦P/      (Define footers to display page numbers
SR   P 0             as register P which is automatically
SR   C 0             increased by 1 as each page is printed;
SR   S 0             also set chapter, section and sub-
SR   T 0             section heading numbers to zero.)
```

Chapter heading macro:

```
DM   CH              (Define chapter heading macro.)
SR   C ¦C+1          (Increase chapter number by 1.)
SR   S 0             (Set section number to Ø.)
SR   T 0             (Set sub-section number to Ø.)
PE                   (Eject page – so chapter begins on new
                     page.)
LJ   *CHAPTER ¦C*    (Print chapter heading.)
CE   *@0*            (Print chapter title, bold, centered.)
EM
```

Section heading macro:

```
DM   SE              (Define section heading macro.)
SR   S ¦S+1          (Increase section number by 1.)
SR   T 0             (Set sub-section number to Ø.)
PE   5               (Eject page if within five lines of
                     bottom.)
```

```
LJ    *¦C.¦S  @Ø        (Print chapter and section numbers,
EM                      and section heading in bold.)
```

Sub-section heading macro:

```
DM    SS                (Define sub-section heading macro.)
SR    T  ¦T+1           (Increase sub-section number by 1.)
PE    3                 (Eject page if within three lines of
                        bottom.)
LJ    *¦C.¦S.¦T  @Ø     (Print chapter, section and sub-section
EM                      numbers, and sub-heading in bold.)
```

When the text is written in VIEW, the footers appear automatically, showing the page numbers, and the numbers in the chapter, section and sub-section headings are provided by the macros, the words of these headings being parameters to the macros, as follows.

```
CH   Chapter title
SE   Section heading
SS   Sub-section heading
     Text text text text text text text text text text text text
     text text text text text text text text text text text text
     text text text text text text text text text text text text

SE   Section heading
     Text text text text text text text text text text text text
     text text text text text text text text text text text text
     text text text text text text text text text text text text
```

If the macros above are made into a file called 'MACRO' and the text into a file called 'TEXT' then the command

```
PRINT  MACRO  TEXT RETURN
```

will reproduce the text shown earlier in this chapter. You will notice, however, that it first prints out a page which is blank apart from the footer, `Page  Ø`. If you can work out why it does this you will have no difficulty in understanding macros.

The reason is that the chapter heading macro causes each chapter to be printed on a new page by doing a page eject. The setting-up sequence sets the page number P to Ø, and the chapter heading macro then ejects page Ø, and P is incremented to 1. This gives the correct starting page and chapter (chapter 1 page 1), but also gives a blank page Ø.

11 Change... replace ... search

Another of those boring and time-consuming tasks which seems to attract errors is going through a document to change a word every time it occurs. For example, you may already have typed a letter which sets out an agreement with a Mr Q R Smith, and you now want to send an identical letter to make a similar agreement with a Mr H I Jones. Or maybe you are typing a text about computers in which you have used the spelling 'disc' – only to find that your editor absolutely insists on the spelling 'disk'.

With traditional typing, making a change like this must count as the most boring of all possible jobs, but with word processing it could not be simpler. In the last example all you would do would be to switch to command mode and type:

```
CHANGE disc disk RETURN
```

and the word would be changed automatically throughout the whole text. Try a few examples for yourself.

Notice how the **CHANGE** command copies the case of the word to be changed. So with the command above, for example, 'DISC' would be changed to 'DISK' and 'Disc' into 'Disk'.

Naturally you can use this not only to correct spelling but to change the terminology itself. For example if you are writing about antiques and mention 'grandfather clocks', you may find your editor insisting that 'long-case clocks' is the correct horological term. No problem:

```
CHANGE grandfather long-case RETURN
```

and again 'Grandfather' will become 'Long-case' too.

The way in which VIEW copies the case of the original can be very useful, but it can also prevent you from altering the case if you want to. For example you may be producing a report about an antibiotic called 'erythromycin'. You have nearly finished when you find that this is a proprietary name, and should be written 'Erythromycin'.

To make a correction like this you need to switch off VIEW's facility to copy the case of the original. This facility is known as 'folding', and you can switch it off by using the command FOLD (in command mode) with the number Ø after it:

```
FOLD Ø RETURN
```

After this you can issue the instruction:

```
CHANGE erythromycin Erythromycin RETURN
```

and the change will be made. To turn folding on again, replace the Ø with a 1 – and if you forget whether folding is on or off, just type

```
FOLD RETURN
```

and VIEW will tell you.

REPLACE

The REPLACE command is used when you want to change some of the words only, and want to make a decision on each one. Suppose you refer to a report in your text, giving the author as 'Green'. You subsequently discover that the author was 'Stephenson', so naturally you use the CHANGE command. To your dismay you later discover that you have also changed 'the Green Hall' to 'the Stephenson Hall'.

In cases like this you should use the REPLACE command instead of CHANGE. It allows you to make a separate decision on each occurrence of the word. The command is given in much the same way as CHANGE:

```
REPLACE Green Stephenson RETURN
```

The system switches to text mode and the first occurrence of 'Green' is signalled by the cursor being placed on the first letter. If you do not want that word replaced, press N. If you want it replaced press Y. VIEW then moves on to the next occurrence of the word, until you have made your decision on them all.

Folding works with REPLACE in the same way as it does with CHANGE.

SEARCH

If you want to find a particular word but do not want to change it, or cannot predict how you will change it until you see it, the most useful command is SEARCH. The command is given in the same way as the commands to change text:

SEARCH Green **RETURN**

The system changes to text mode and the cursor rests on the first occurrence of the word named. To find the next press NEXT MATCH (**CTRL** **f1**).

So what happens when you search for a word which you know is there, but the SEARCH command cannot find it? Perhaps you have mistyped it? – in which case the SEARCH command will never find it. Here again VIEW comes to the rescue with the 'wild search' facility.

All you have to do is to operate the search while substituting a '?' for the characters which may be wrong. So if you want to find 'Erythromycin' but are not sure how it is spelt, you will find it quite quickly if you ask for 'Erythro?????'.

With commoner words it will of course find other words of similar spelling, but you can quickly move on using NEXT MATCH.

Even **RETURN** and **TAB** can be specified in your searches, using the vertical bar and the tilde (~) respectively. More detail on this later.

Limited searching and changing

You can cause the commands SEARCH, CHANGE and REPLACE to act on a limited area of the text if you set markers 1 and 2 before and after the text concerned and then name these markers in the command:

CHANGE disc disk 1 2 **RETURN**

This will change 'disc' to 'disk' between the markers only.

Finding and changing phrases

The space between the words of a CHANGE command has a definite function: it signals to VIEW that you want the word before the space

changed into the word after it. To replace one group of words with another, therefore, requires a slightly different method.

The method is to enclose the phrases within slashes, like this:

```
CHANGE/read only memory/random access
memory/ RETURN
```

The space or slash in **CHANGE** and similar commands is known as a 'delimiter' since its function is to show where one thing ends and another begins. The first character after **CHANGE** indicates to VIEW which delimiter is being used.

12 Special facilities

COUNT

To find out the number of words in your text, switch to command mode and type

COUNT **RETURN**

If you want to know the number of words in part of the text only, set markers 1 and 2 before and after the part concerned and type:

COUNT 1 2 **RETURN**

It is important, however, to recognize what VIEW is doing when it counts words. It would be truer to say it was counting the spaces between the words. VIEW cannot read, so the only way it can discover a word is by the fact that words have spaces each side of them. So if you have typed 'WORD' it will recognize it as one word. If you have typed 'W O R D' VIEW will count it as four words, but if you are aware of how VIEW goes about its task, COUNT is a very useful facility.

FORMAT

This is similar in its effect to the FORMAT BLOCK command which formats paragraphs. FORMAT is used in command mode and can format all the text in memory.

As such it must be used with caution. It may be tempting to decide to use a narrower column, for example, and apply this throughout the whole text. All you have to do is to set a shorter ruler, go into command mode and type:

FORMAT **RETURN**

However, you should check the text carefully before using FORMAT. Some parts of it may be unsuitable for formatting – for example where you have used a narrow column and inserted side headings; or where you have a table which is not protected from the effects of formatting.

To limit the effect of the FORMAT command, you can set markers each side of the text to be formatted and type:

```
FORMAT 1 2 RETURN
```

For more information on the effects of TAB on formatting, see chapter 5 of this book.

Editing BASIC programs

It is possible to use VIEW to edit BASIC programs. While you may not often wish to do this – since it is quite possible to edit them in BASIC itself – it is sometimes useful to be able to place a part of a program within a VIEW text. For example you may have invented a new BASIC routine and want to write an article describing how it works. It would be very boring to have to type all the program lines into a VIEW file, when you have already got them in a BASIC file on your disk. It could also lead to a lot of mistakes.

The method of getting BASIC files into VIEW and back into BASIC again is as follows.

To place a BASIC program in a file which VIEW can read

Type: *BASIC RETURN (To get into BASIC.)

Place the disk containing the BASIC program in the drive.

Type: LOAD filename RETURN (To load the program.)

Type: *SPOOL newfile RETURN (To create a new file.)

Type: LIST RETURN (To put the program into it.)

Type: *SPOOL RETURN (To close the new file.)

To read the program into VIEW and edit it

Type: *WORD RETURN (To get into VIEW.)

Type: NEW RETURN (To clear text mode.)

Type: READ newfile RETURN (To read in the file.)

You can now edit the program and SAVE it back in 'newfile' or in another text file if you wish.

To use the edited version as a BASIC program

Type: `*BASIC` **RETURN** (To get into BASIC.)

Type: `*EXEC newfile` **RETURN**

The program is now in memory and can be listed and run in BASIC.

The point of all this routine is that VIEW cannot use BASIC files, so you have to create a file with the program in it which VIEW can use. This is done by using the `*SPOOL` command, which makes a file out of whatever is on the screen – and the program is placed on the screen by listing it.

The `*EXEC` command has the opposite function, of making a file which BASIC can use out of a VIEW text – which must of course be in the correct form for a BASIC program.

One point is worth stressing: always use `READ`, not `LOAD`, or you will have a completely unusable column of text in VIEW.

13 Continuous processing

Up till now, we have been thinking in terms of composing text, putting it on to disk, and retrieving it one file at a time, using SAVE and LOAD, or perhaps READ if we want to add material to a file we are already editing.

This is quite satisfactory for short items, but suppose you are processing a long report and want to go through it from beginning to end, correcting and editing. You will find yourself constantly interrupting your editing to LOAD and SAVE files.

A convenient way round this problem is offered by the EDIT facility, and when you have become accustomed to using VIEW you may find yourself using this as much as the separate SAVE and LOAD commands.

What happens in principle is that the computer reads in material from one file, holds it in memory while you edit it, and saves it in another file, bringing in more material from the first file automatically. The process continues until you have edited all the material and passed it on to the second file.

Obviously if you are going to use the EDIT system, you have to make sure that you have plenty of memory left on your disk before you start, or you will get a Can't Extend message and you will be left with a computer full of edited material and nowhere to put it. With care, however, it can be an effective way of processing files which contain much more material than will fit into the computer's memory.

The EDIT method

If you have material in the computer's memory which you wish to preserve, SAVE it now. We are about to compact the disk so as to use disk memory more efficiently, but compacting normally destroys any text in the computer's memory.

Check the files on your disk. Type *. RETURN or *CAT RETURN. Are there any files you want to delete? If so type *DELETE filename RETURN.

Compact your disk. Compacting rearranges the files in the most economical manner. Type `*COMPACT` **RETURN**.

These first three instructions are intended to leave your disk with its programs arranged so as to leave the maximum memory available for the `EDIT` process. Remember that your editing may result in a much larger file than you started with.

You are now ready to start the `EDIT` procedure, by reading in your file and giving a name to the file to which the text you are processing will go. Type

```
EDIT file-in file-out RETURN
```

('File-in' is the original file, and 'file-out' the destination file.)

VIEW reads in text from 'file-in'. You process it. When you are ready to go on to the next batch type

```
MORE RETURN
```

The text you have processed is written into your 'file-out' and new text is read in from your 'file-in'.

When you wish to stop editing, even if you have not finished the document, type

```
FINISH RETURN
```

and the text in memory is put into your 'file-out'. Any unread text left in 'file-in' is read and transferred to 'file-out'.

You are now left with two files: the original file ('file-in') and the destination file ('file-out'). If you no longer need 'file-in' type

```
*DELETE file-in RETURN
```

The command `QUIT` is an alternative to `FINISH`, which simply abandons editing, leaving 'file-in' intact but 'file-out' incomplete. You should delete 'file-out'.

You should be particularly careful about naming your files if you are to use `EDIT`. It is easy enough to get confused with normal saving and loading, but using two files at the same time can be tricky if you are

careless about naming them. You can have 'file-in' and 'file-out' on different disk drives by specifying the drive in the name, eg

```
EDIT :0.source :2.dest RETURN
```

See the Disk Filing System User Guide for details.

Finishing

Note the procedure for finishing in the instructions. The point is that you can end your editing session in three ways. You can keep on asking for `MORE` until you finish the job. You can type

```
FINISH RETURN
```

if you have done a good deal of the job and want to preserve your text in a half-edited condition, to continue editing it later. `FINISH` leaves you with all the edited text in 'file-out' along with the rest of the text which you have not yet edited. In your next session you have only to keep on asking for `MORE` until you come to where you left off.

If on the other hand you have been interrupted when you have just started to edit, and what you have done so far is not worth saving, you do not want to wait while the computer goes through all the process of transferring unedited text from 'file-in' to 'file-out', as it would have to do if you used `FINISH`. Instead you can use the command `QUIT`, which simply abandons the operation, so that you can start again in your next session with the same 'file-in'.

One final refinement in continuous processing. When you are in the middle of editing, you may wish to transfer part of the text you are working on to 'file-out' in order to get some more out of 'file-in', perhaps to compare or transfer parts of it.

You can do this by setting marker 1 at the point up to which text should be transferred to 'file-out'. Then type `MORE` RETURN. The first part of the text will then go into 'file-out', and more material will arrive from 'file-in'.

VIEW Guide

Introduction

Commands

Immediate commands

These are used in text mode and take effect immediately they are given. Most immediate commands are given through the red function keys at the top of the keyboard.

Stored commands

These are also used in text mode, but have no immediate effect, except for the placing of the appropriate code letters in the left margin. Stored commands take effect only when the document is being printed.

Command mode commands

These are given in command mode and control general functions such as recording and printing.

Presentation of commands

Commands shown reversed out white on black, eg **RETURN**, mean 'press the key named'; commands in ordinary upper case refer you to the prompt card which shows the **f** key to be pressed; curved parentheses mean 'type the information described within the parentheses'; other words and numbers in computer typeface should be typed exactly as shown – *including spaces.*

Examples of presentation

EDIT COMMAND `TM` **RETURN** `(number of lines)`

means press the EDIT COMMAND key (**SHIFT** and **f8**); type `T`; type `M`; press **RETURN**; type a number.

`CHANGE (old word) (new word) 1 2` **RETURN**

means type the word `CHANGE`; type a space; type the old word; type a space; type the new word; type a space; type `1`; type a space; type `2`; press **RETURN**.

Useful hints

The BREAK key

Pressing the **BREAK** key leaves the system in command mode including the `No text` message, which means that your text has vanished. To recover text lost in this way type `OLD` and press **RETURN**.

Screen placing

You may find that the screen image is too high for you to read the top line. To lower the image type `*TV255` **RETURN** and `MODE3` **RETURN**. Note that `*TV` commands are only effective after a `MODE` command is given.

BASIC

To use BASIC instead of VIEW, type `*BASIC` **RETURN**. To change back to VIEW type `*WORD` **RETURN**.

1 Start-up

To get into VIEW type `*WORD` **RETURN**.

Command mode

```
VIEW A2.1
No text
Editing No File
Screen Mode 7
Printer default
=>
```

`VIEW A2.1` is the version of VIEW you are using.

`No text` means that no text has yet been typed in or loaded from tape or disk.

`Editing No File` means that no file is being edited. If you `LOAD` a file from disk or tape, the display changes to `Editing (filename)`.

`Screen Mode 7` gives you 20 lines of 34 characters. For more information on modes, see page 230.

`Printer default` means that the built-in printer driver (ie a program to operate the printer) is operating. For more information see page 240.

`=>` indicates where commands are typed in. Note that at start-up 'caps lock' is on. To turn it off press the **CAPS LOCK** key.

Changing modes

To switch between command mode and text mode press **ESCAPE**. Note that you cannot switch to text mode if there is no text. Either load text from disk or tape or type `NEW` and press **RETURN**. The display will then change from `No text` to `Bytes free...`

`Bytes free...` shows the amount of memory free for you to use. As a rough guide one byte corresponds to one character (ie one letter, number, sign or space).

Text mode

```
FJ .......*.......*.......*.......*.<
_
**********************************
```

The ruler controls the width and tabbing of the column of text under it.

< is the right margin.

* is a TAB stop.

J means justified. When J is shown at top left on the screen, the spaces in a typed line are adjusted to make all lines of equal length.

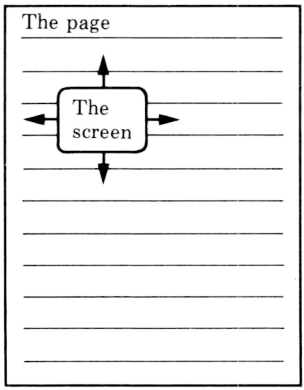
_ is the cursor.

F means format. When F is shown at top left on the screen, words which overshoot the right margin are transferred in entirety to the next line.

The row of asterisks marks the end of the column of text.

If you hold down **SHIFT** and press an arrow key the effect is:

- Horizontal arrow keys: jump to the first letter of the next word forward or back, throughout the text.
- Vertical arrow keys: jump the height of the screen up or down.



The screen operates as if it were a window looking on to a much larger page. Use the arrow keys to move the screen over the page.

2 Immediate commands

Moving the cursor

TOP OF TEXT moves the cursor to the first character of the text (even if that happens to be part of a user defined ruler).

BOTTOM OF TEXT moves the cursor to the space after the last character on the lowest line, above the row of asterisks.

BEGINNING OF LINE moves the cursor to the left of the current line.

END OF LINE moves the cursor to the space after the last character in the current line.

GO TO MARKER moves the cursor to a marker (1 to 6) already set with the SET MARKER command. To set markers, see page 2Ø5. Press GO TO MARKER followed by the number of the marker required.

Under normal operation the arrow keys (to the right of the keyboard) move the cursor one space in the direction indicated. When used with the **SHIFT** key the horizontal arrow keys cause the cursor to jump from word to word and the vertical arrow keys cause it to jump the height of the screen, up or down.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Lines

INSERT LINE inserts a blank line above the line which the cursor is on.

DELETE LINE deletes the line which the cursor is on.

SPLIT LINE splits off the part of the line to the right of the cursor (including the character above the cursor) and places it on a new line inserted immediately below. This command will not work if there is a stored command on the line concerned.

CONCATENATE LINES joins the line below on to the line which the cursor is on and closes up the text below. To concatenate, place the cursor anywhere on the upper of the two lines. Then press CONCATENATE LINES. This command will not work if there is a stored command on either of the lines concerned. If the two lines together total more than 132 characters, the lines will not be concatenated and the computer will bleep.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Deletion

DELETE CHARACTER deletes the character above the cursor and closes up the gap.

DELETE END OF LINE deletes the character above the cursor and all characters to the right of the cursor on that line.

DELETE UP TO CHARACTER deletes the character above the cursor and all characters to the right of it on that line *up to a specified character.*

To use the command – place the cursor at the start of the deletion. Then decide which character should end it (or insert a special one for the purpose).

Then press DELETE UP TO CHARACTER followed by that character. If the character occurs several times together, the system will delete all of them.

DELETE LINE deletes the line which the cursor is on.

DELETE BLOCK – see page 2Ø6.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Insertion and swapping case

INSERT CHARACTER inserts a space at the cursor, moving existing text to the right. (For example: `PRNT` becomes `PR_NT`).

INSERT MODE When INSERT MODE is pressed, the letter `I` appears at the top left of the screen. Any characters typed in will then appear at the cursor, and text will move right to accommodate them. To cancel insert mode, press the key again.

INSERT LINE inserts a blank line above the line which the cursor is on.

SWAP CASE changes the case of the character above the cursor (ie capitals to small, or small to capitals), and moves the cursor right.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Markers

SET MARKER Markers may be set singly or in pairs to define a block of text to be moved, copied or deleted. They are identified by numbers 1 to 6. Markers 1 and 2 are shown as inverse characters in MODES Ø to 6, and as a square in MODE 7. Markers 3 to 6 are not displayed, but you can find their positions by using the GO TO MARKER command (**SHIFT** **f6** followed by the marker number), for example

GO TO MARKER 3

A message in command mode indicates which markers have been set.

To set a marker:

- Move the cursor to the place where you want the marker.
- Press SET MARKER. (**MK** appears at top left on the screen.)
- Press a number 1 to 6.

To clear markers 1 and 2:

- Press **ESCAPE** to enter command mode.
- Type `CLEAR` **RETURN**.

Markers 3 to 6 cannot be cleared, but have no effect on the text.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Block operations

To delete a block of text

Set marker 1 at the first character of the text to be deleted and marker 2 at the space after the last character.

Press DELETE BLOCK (**CTRL** **f0** , not the black **DELETE** key).

To move a block of text

Set marker 1 at the first character of the text to be moved and marker 2 at the space after the last character.

Place the cursor on the line you want the text to move to. This may be anywhere except between the two markers.

Press MOVE BLOCK (**SHIFT** **f0**).

To copy a block of text

Set marker 1 at the first character of the text to be copied and marker 2 at the space after the last character.

Place the cursor on the line where you want the copy to appear. This may be anywhere except between the two markers.

Press **COPY** (at bottom right of the keyboard).

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

The ruler

```
.. .......*.......*.......*.......*.......*.......*.......*.......*.......*.<
```

This is the default ruler for MODE 3; it is 74 characters wide. ('Default' means that it is what the system gives you unless you request something different.) The default ruler for the 4Ø character line is 34 characters wide.

The dots indicate spaces. The * characters are TAB stops. A letter b in the ruler activates a bell-stop. The < is the right margin stop, marking the last character position in the line. You may also set a > for a left margin stop. You may use up to 128 rulers in a document.

The system obeys the last ruler *above the line which the cursor is on.*

For the effect of the ruler, see 'formatting and justification' (page 2Ø9).

To obtain a default ruler for a given text mode – press DEFAULT RULER.

To copy the current ruler – press **SHIFT** and **COPY**.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

To alter a ruler – place the cursor on it and use the normal commands, such as INSERT CHARACTER to lengthen it, and DELETE CHARACTER to shorten it. Similarly you may delete asterisks and type new ones in to adjust TABs. You may delete margins and enter new ones using > (left) and < (right).

To make up your own ruler – place the cursor on the line where you want the ruler, and press MARK AS RULER (**CTRL** **f8**). Two dots (. .) appear in the left margin and the line becomes a ruler. To operate TABs and formatting, you have to include * and > and < in your line. You can also include character b which causes a bleep when the cursor passes it. *Any spare space in the line you should fill with dots.*

The ruler may be 132 characters wide in any mode. How many characters are displayed on the screen at any time depends on the mode – 74 characters are displayed in MODE 3.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Formatting and justification

FORMAT MODE Formatting relates the length of text lines to the current ruler.

Press FORMAT MODE (**CTRL** **f2**) to enter format mode (F is displayed at top left). Press FORMAT MODE again to leave format mode (nothing is displayed).

In format mode, when text is typed in, any word which overflows the right margin is transferred whole to the following line. In effect, switching off formatting releases the right margin only.

JUSTIFY MODE adjusts the spaces in the lines so that all lines which reach the right margin are the same length, ie the right margin is justified.

Press JUSTIFY MODE (**CTRL** **f3**) to turn on (J is displayed at top left). Press JUSTIFY MODE again to turn off (nothing is displayed).

When both F (format mode) and J (justify mode) are displayed and text is typed in, any word which overflows the right margin is transferred whole to the following line, and the spaces in the upper line are adjusted to justify the lines.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Formatting and margins

FORMAT BLOCK To reformat a paragraph, place the cursor anywhere on the top line and press FORMAT BLOCK (**f0**). This will work only if VIEW is in format mode, and the text will be justified or not according to whether the system is in justify mode. The formatting will continue down the paragraph until a blank line is encountered. Do not attempt to format a paragraph in which more than one TAB has been used.

FORMAT BLOCK can be used to alter the width of a column of text. To do this, first change the ruler above the text. Then use FORMAT BLOCK.

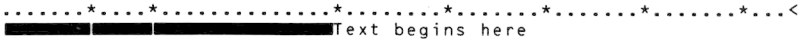
RELEASE MARGINS (**SHIFT** **f2**) releases both margin stops and turns off formatting. Press again to restore them. You can use this command to enter text to the left of the left margin stop by releasing margins, typing in the text, and restoring them again. This can be useful for side headings in a report.

To format the whole of the text in memory, or of several paragraphs, see page 236.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

TAB stops and characters

Pressing TAB appears simply to move the cursor along to the space below the next asterisk on the current ruler. In fact, each time TAB is pressed, an invisible TAB character is inserted into the text like this:



(Three TAB characters)

TAB characters vary in length according to the setting of the asterisk TAB stops.

To insert TAB characters in existing text (ie to move the text one TAB right):

- Place the cursor to the left of the text.
- Press TAB.

To delete the rightmost TAB character in existing text (ie to move the text one TAB left):

- Place the cursor to the left of the text.
- Press DELETE CHARACTER and the whole TAB character is deleted.

TAB characters and formatting

VIEW will not format a line starting with spaces or a single TAB. This can be used to protect tables from the effects of formatting commands.

If you have used more than one TAB character in a line, you should not attempt to use the FORMAT BLOCK command, since it will reformat the TABs.

Moving the cursor to a point within a TAB character

While the character is in place, this cannot be done. If you use the arrow keys, the cursor jumps from one TAB character to the next. You should use the arrow keys to move the cursor up to the beginning of the TAB character, then press the space bar to insert as many spaces as you need to get to the point you require.

Note: If you have inserted TAB characters into a line, and then type over the line again, as you approach the first TAB character, VIEW moves it and the rest of the line that follows to the right one TAB space. If you no longer require the TAB character, you can remove it by pressing DELETE CHARACTER, and the text will close up to the cursor.

Alternative TAB

The command `FIELD` allows you to give another character the function of the TAB.

Switch to command mode by pressing **ESCAPE**. Then type

```
FIELD (ASCII number of the required
character) RETURN
```

For example `FIELD 32` **RETURN** turns the space bar into a TAB key. This is useful when typing columns of figures.

3 Stored commands

Stored commands are entered in the left margin, to be used later when the document is printed. Some are used on their own, and some with a number or text beside them.

Using a stored command on its own, type:

EDIT COMMAND `(command code)` **RETURN**

For example:

EDIT COMMAND `PE` **RETURN**

`PE` stands for page eject.

Using a stored command followed by characters, type:

EDIT COMMAND `(command code)` **RETURN** `(character)`

For example, to modify the effect of a command:

EDIT COMMAND `PE` **RETURN** `10`

(This means page eject if there are ten lines or fewer remaining on the page.)

To put the command into effect, type:

EDIT COMMAND `PL` **RETURN** `45`

(This means 'set a page length of 45 lines'.)

To place text controlled by the command, type:

EDIT COMMAND `CE` **RETURN** `Title`

(This means 'center the word `Title` between the margins'.)

To delete a command place the cursor on the same line as the command and press DELETE COMMAND.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Lines and page length

Center line
Centers the text that follows this command in relation to the left and right margins.
Code: CE
Use EDIT COMMAND CE **RETURN** (text to be centered)

Right justify
Aligns the text that follows this command with the right margin.
Code: RJ
Use EDIT COMMAND RJ **RETURN** (text to be ranged right)

Left justify
Can be used to align text with the left margin and to print out the values of number registers. For an explanation of this, see page 229.
Code: LJ

Page length
The system assumes a page length of 66 lines, unless told otherwise.
Code: PL
Use EDIT COMMAND PL **RETURN** (number of lines)

The maximum page length that may be set is 255 lines.

Line spacing
The system assumes solid text.
Code to insert blank lines for double spacing etc: LS
Use EDIT COMMAND LS **RETURN** (number of lines spacing)

Ejecting pages

Page eject
Causes the printer to eject the page.
Code: PE

To eject the page at the point where the command is entered:
EDIT COMMAND PE **RETURN**

To eject the page when the number of lines specified or fewer remain on the page:
EDIT COMMAND PE **RETURN** (number of lines)

Eject till odd or even page number
Codes: OP and EP

These commands make it possible, for example, to ensure that all chapters start on a right-hand page. The method is first to ensure that the page counter (register P) starts at 1 for the first chapter (see page 229 for registers), and then at the end of each chapter to use code OP instead of the usual PE. If the page (register P) happens to be even, the page ejects as usual. If it is odd, there is an additional page eject, so that the next chapter can start on an odd page.

Formally either OP or EP causes a page eject, plus an additional page eject if the page number after that ejection is not odd (with OP set) or even (with EP set).

The codes are set with:
EDIT COMMAND OP **RETURN**
EDIT COMMAND EP **RETURN**

Page layout

VIEW's standard default page layout is shown below. At the top of the page is a top margin of four lines. Below it is the header line which can have *one* or *all* of a left component, a central component and a right component. Below the header is a header margin – the default for this is also four lines.

Similarly there is a bottom margin and a footer margin – of four lines each. The default page length is 66 lines. This default layout is designed to conform to the standard format for business letters.

Example of page layout

Overleaf is a sample page showing the text from the previous page printed out using different page layout commands from the default settings. The list of stored commands at the top of the file would be as follows:

```
TM 3
DH /Creative Graphics//Final draft/
HM 3
PL 30
DF //Volume 1, page ¦P//
FM 3
BM 3
```

The printed page looks like this:

Creative Graphics Final draft

Lissajoux figures

Lissajoux figures are fascinating patterns that can form the basis for many weird and wonderful programs. The method for drawing Lissajoux figures is similar to the polar-coordinate method for drawing a circle. For the circle, the angle from which the x- and y-coordinates are derived is the same. Different Lissajoux figures are obtained when these angles are out of phase. The following program draws a different Lissajoux figure each time the Space Bar is pressed.

There are many ways in which the basic Lissajoux patterns can be enhanced. Here is a program that uses straight lines to join the points that trace

Volume 1, page 1

Creative Graphics Final draft

out two intermeshing figures. The pattern obtained depends on the random numbers chosen at lines 50 and 60.

Volume 1, page 2

Page layout commands

Top margin
Code to adjust the top margin: TM
(The top margin is between the top of the page and the header.)
Use EDIT COMMAND TM RETURN (number of lines)

Header margin
Code to adjust the header margin: HM
(The header margin is between the header and the text.)
Use EDIT COMMAND HM RETURN (number of lines)

Bottom margin
Code to adjust the bottom margin: BM
Use EDIT COMMAND BM RETURN (number of lines)

Footer margin
Code to adjust footer margin: FM
Use EDIT COMMAND FM RETURN (number of lines)

Left margin
A general instruction can be entered which sets the left margin for the whole document.
Code to adjust the left margin: LM
Use EDIT COMMAND LM RETURN (number of spaces)

Note that the LM command controls printing only, and has nothing to do with the margin stops as set in the ruler. VIEW assumes no left margin at all on printing unless this is altered with LM.

Headers and footers

You may define headers and footers, so that every page of your document has the same headers and footers printed on it automatically. Headers and footers have left, center and right components. The maximum length of a header or a footer is 63 characters.

Defining headers and footers

Define header
Code: DH
Use EDIT COMMAND DH **RETURN** /left component/center component/right component/

Define footer
Code: DF
Use EDIT COMMAND DF **RETURN** /left component/center component/right component/

Once headers or footers are defined they will appear automatically on all pages unless you turn them off.

To omit a component of a header or footer, type two slashes together. For example:

EDIT COMMAND DH **RETURN** //Project 5//
(This has a center component only.)

Note: Although a slash (/) is shown above for separating components, you may use any punctuation characters, as long as you use the same one all the way through. It will be treated as a spacer only.

Turning headers off or on
Code: HE
Use EDIT COMMAND HE **RETURN** 0
To turn them on again, replace 0 with 1. (The default is on.)

Turning footers off or on
Code: FO
Use EDIT COMMAND FO **RETURN** 0
To turn them on again, replace 0 with 1. (The default is on.)
Note that with headers and footers turned off VIEW prints a single blank line.

Two-sided documents

When processing books it is often useful to swap over the parts of headers and footers on alternate pages. For example, page numbers normally appear on the outer margins, and chapter and book titles are often arranged like this:

```
Chapter title          Book title      Book title          Chapter title
```

Code: TS
Use EDIT COMMAND TS **RETURN** 1
To cancel replace 1 with 0 (The default is off.)

When using this facility, the left component and right component defined by DH or DF will exchange on each page eject PE.

You can also use this command to increase the left margin on right-hand pages to make room for the binding.
Use EDIT COMMAND TS **RETURN** 1 (increased spaces in margin)
For example: TS 1 15 would turn on the two-sided document facility with 15 extra spaces in the margin.

Macros

If you need to print any block of text more than once, VIEW allows you to define it as a macro, giving it a two-letter name. You can then use it, as often as you like by entering its name in the margin, exactly as you would a stored command.

Codes to define macros: DM and EM (ie 'define macro' and 'end macro')

To define a macro

Type EDIT COMMAND DM **RETURN** AB
(AB represents a two-letter name for that macro. You may use any two-letter name which is not a command code already.)

Type in the macro, using text and/or commands.

Type EDIT COMMAND EM **RETURN** to end the macro.

To use a macro

Place your chosen two-letter name for that macro in the margin, as if it were a normal stored command. Names of macros should not duplicate those of stored commands; ie use any combination of two letters which is not in the list of stored commands.

Use EDIT COMMAND AB **RETURN**

Note: You are not allowed to use a macro within a macro.

Number registers (see page 229) may be used in macros, with such commands as RJ, CE and LJ.

Macros with parameters

A powerful feature of VIEW's macros is that you can modify them each time you use them. In effect you leave gaps in them and fill in the gaps when printing. In this way it is possible to construct a standard text and print it out many times. The method is as follows.

The macro is defined as on page 222, but some words in it are replaced by up to ten symbols @Ø, @1, @2 . . . @9.

Each time the macro is used, these parameters are supplied in the form of a list of words accompanying the name of the macro. The parameters for any printing of a macro must be separated from each other by commas. If you wish to include a comma within a parameter you must enclose the parameter within angle parentheses.

A typical print instruction for a macro (named AB) with parameters would be:

```
AB 7th June,<Dear Mr Weavelburg,>,interest,16%,3rd July
```

Macro example

Suppose we wish to send a letter to applicants for a course. Below is a typical letter of this kind. Following it is the equivalent macro, with codes inserted to mark the positions the parameters will occupy.

```
Ms E Schulz                Sept 15 1983
2760 E Skelly Drive
Waltham
MA 02356

Dear Ms Schulz

Thank you for your application.  We
confirm your enrolment for the
Intermediate Course F.  Your student
number is 1793.

You will receive further details in
a few days.  Meantime, will you
please send the acceptance payment,
as shown in our brochure, of $42.50.

If you have any queries please
address them to Dept 12.

Yours sincerely
```

```
DM LL
@0                                   Sept @1 1983
@2
@3
@4

Dear @5

Thank you for your application.  We
confirm your enrolment for the
Intermediate Course @6.  Your student
number is @7.

You will receive further details in
a few days.  Meantime, will you
please send the acceptance payment,
as  shown in our brochure, of @8.

If you have any queries please
address them to Dept @9.

Yours sincerely

EM
```

Calling the macro

The macro is named as LL by the **DM** stored command, and it is ended by the **EM** command. By itself it will not print out, but when its name is entered in the margin like a stored command with a line of parameters beside it (this is referred to as 'calling a macro'), any number of individualized letters can be printed.

The macro call for the letter above would be:

```
LL Ms E Schulz,15,2760 E Skelly Drive,Waltham,MA 02356,Ms Schulz,F,1793,$42.50,12
```

Another parameter line for this letter is shown in the following macro call.

```
LL Mr P Flood,30,<68, Rutherford Blvd>,Beachwood,OH 45341,Mr Flood,E201,736,$60.50,22
```

Each of the items enclosed by commas corresponds in order to each of @Ø, @1, @2, @3, @4, @5, @6, @7, @8 and @9.

Note the angle parentheses in the last example, enclosing a parameter which includes a comma. Note also that all parameters must be on the same line, the maximum length for a line in VIEW being 132 characters.

If you wish to ignore one of the parameters, use commas as appropriate. For example, a one-line address can be dealt with as follows:

```
LL Mrs C Martin,6,Room 112,,,Mrs Martin,C,553,$22.50,4B
```

Highlight 1 and 2

The highlight commands 1 and 2 mark parts of the text which are to be printed in a special way.

The printer is controlled by a printer driver program – either the default printer driver contained in VIEW or a driver designed to run a specific printer. Specific printer drivers are known by the name of the printer (eg RICOH) and have to be loaded from disk or tape.

The highlight commands send codes to the printer driver, which converts them into instructions for the printer. In the default mode, HIGHLIGHT 1 sends code 128 and HIGHLIGHT 2 sends code 129. Provided the correct printer driver program has been loaded, and the printer is capable of responding to the instructions, HIGHLIGHT 1 results in underlined type and HIGHLIGHT 2 in bold type.

Some printers however are capable of special effects other than underlining and bold – for example special typefaces or superscripts. If a printer driver program has been written for such a printer, other codes will have been assigned to each of these effects.

In order to use the highlight commands to send these alternative codes to the printer driver it is necessary to switch the highlight commands temporarily from codes 128 and 129 to the other numbers which are built into the driver. This is done with the HT command, as described below.

The advantage of this system over sending control characters directly to the printer is that you can print your text on different printers by simply changing the printer driver, rather than having to redo all the control codes throughout the text.

CTRL ▶	DELETE BLOCK	NEXT MATCH	FORMAT MODE	JUSTIFY MODE	INSERT MODE	DEFAULT RULER	SPLIT LINE	CONCATENATE LINES	MARK AS RULER	
SHIFT ▶	MOVE BLOCK	SWAP CASE	RELEASE MARGINS	DELETE UP TO CHARACTER	HIGHLIGHT 1	HIGHLIGHT 2	GO TO MARKER	SET MARKER	EDIT COMMAND	DELETE COMMAND
	FORMAT BLOCK	TOP OF TEXT	BOTTOM OF TEXT	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	INSERT LINE	DELETE LINE	INSERT CHARACTER	DELETE CHARACTER

Marking highlight 1

Place the cursor under the first character of the text concerned and press HIGHLIGHT 1. The text will move right and an underline will appear.

Move the cursor to the space after the last character of the text concerned and press HIGHLIGHT 1 again. A second underline will appear and if there is any text to the right it will move to make room for the underline.

When printed the text between the underlines will be in the highlight 1 mode, ie unless the effect has been changed with the HT command, it will be underlined.

Marking highlight 2

Follow the same procedure as for highlight 1. Instead of an underline the marker in the text is an asterisk. In the default mode this will result in bold type.

Resetting the highlight commands

Code HT
Use EDIT COMMAND HT RETURN 1 (new code)
EDIT COMMAND HT RETURN 2 (new code)
For example to change highlight 1 from code 128 to 130 use HT 1 130.
To restore the default codes simply use HT again with codes 128 and 129.

Number registers

VIEW has 26 number registers, labelled A to Z. Register P counts pages and L counts lines. Others can be allocated as you wish, and all can be set and reset and can count.

To set or reset, use

EDIT COMMAND `SR` **RETURN** `(register letter) (number)`

For example `SR A 10` sets register A to the value 1Ø.

Instead of a number you may use an expression such as A = A+15, which must be expressed as follows:

```
A ¦A+15
```

(ie A followed by space followed by vertical bar followed by A+15). This means the value of A is increased by 15.

You may print out number register values by including their symbols (A to Z) in the following commands, preceded by the vertical line symbol.

`DH` (define header)
`DF` (define footer)
`CE` (center)
`RJ` (right justify)
`LJ` (left justify)

For example, if you type EDIT COMMAND `DF` **RETURN** `//¦P//` early in the document, each time a footer is printed it will show the page number – the value of P. You can also set the page number to start at whatever number you wish, using the `SR` command.

The command `LJ` (left justify) exists largely for the purpose of printing out register values, since VIEW justifies left by default.

For example `LJ ¦L` would print out the value of register L on the left. You can also include text: `LJ LINE ¦L` would print the value of the L register, which is the line number on the page concerned, preceded by the word 'LINE'. `CE` and `RJ` are used similarly.

Note: In MODE 7 the vertical is displayed as ▮.

4 Command mode commands

Screen modes

VIEW allows you to use any one of eight different screen modes, which allow the following maximum number of displayed characters and lines.

	Characters × Lines
MODE Ø	74 × 25
MODE 1	34 × 25
MODE 2	16 × 25
MODE 3	74 × 22
MODE 4	34 × 25
MODE 5	16 × 25
MODE 6	34 × 22
MODE 7	34 × 2Ø

To switch to another mode, type

`MODE (mode number)` **RETURN**

Different modes require different amounts of memory. If you are running out of memory (for example if you attempt to `SAVE` a file and are told that there is `Not enough memory`) it may be possible to change modes and solve the problem. For example if you run out of memory in this way in MODE 3 you will have ample memory left in MODE 7.

If you request a change of mode and there is not sufficient memory to accommodate your text in the mode you have asked for, the computer will reply `Not enough memory` and leave the mode as it is.

Clearing text from memory

Type `NEW` **RETURN**. All text is cleared but `No text` is not displayed, since a single **RETURN** is left, to enable you to switch immediately to text mode.

Using the disk

`SAVE (filename)` **RETURN** records the named file on to the disk.

If a file of that name is already on the disk, it is overwritten. To record part of a file only, set markers 1 and 2 at the beginning and end of the part you wish to record (to set markers see page 2Ø5). Then type

`SAVE (filename) 1 2` **RETURN**

When a file name appears in the screen message in command mode (`Editing file...`) you can save the text in memory under that file name by typing `SAVE` **RETURN**.

`LOAD (filename)` **RETURN** reads the contents of the file named and places them in the computer's memory. If a file is already present in the memory, it is deleted and replaced by the new file. `LOAD` can be abbreviated to `L.`

`WRITE (filename)` **RETURN** has the same effect as `SAVE (filename)` **RETURN**, except that it is slower.

`READ (filename)` **RETURN** reads the contents of the named file and places them in the computer's memory. If there is text already in memory then the new file will be added to it, if memory space allows.

You can also instruct VIEW to read a file into the middle of the text you are currently writing. To do this mark the place where you want the new file by setting marker 1, and type

`READ (filename) 1` **RETURN**

To check the files on a disk, type

`*.` **RETURN** or `*CAT` **RETURN**

Locking files

To lock, type

```
*ACCESS (filename) L RETURN
```

To unlock, type

```
*ACCESS (filename) RETURN
```

Note: File names can have up to seven characters, letters or numbers. Generally speaking, they should *not include spaces or punctuation marks.*

Editing BASIC programs

VIEW allows you to read programs written in BASIC into memory. This can be convenient both for editing BASIC programs and if you want to quote from BASIC programs within the text you are writing. The one restriction is that the BASIC program may not have lines longer than 132 characters.

To place the BASIC program in a file which VIEW can read

Type: `*BASIC` RETURN	(To get into BASIC.)
Type: `LOAD (filename)` RETURN	(To load the BASIC program.)
Type: `*SPOOL (newfile)` RETURN	(To create a new file.)
Type: `LIST` RETURN	(To get the program into the new file.)
Type: `*SPOOL` RETURN	(To close the new file.)

To read it into VIEW and edit it

Type: `*WORD` **RETURN** (To get into VIEW.)

Type: `NEW` **RETURN** (To clear text mode.)

Type: `READ (newfile)` **RETURN** (Do not use `LOAD`.)

You can now edit the program and `SAVE` it back to the new file or any other.

To use the edited version as a BASIC program

Type: `*BASIC` **RETURN**

Type: `*EXEC (newfile)` **RETURN**

The program is now back in memory and can be listed and run.

Continuous processing (disks only)

This is an alternative to the `LOAD`... process... `SAVE` sequence, and is useful where you have to reprocess a lot of material that is already on disk. It allows you to process files which are larger than the computer's memory.

The command `EDIT` causes the computer to read material from one named file, allows you to edit it, and pass it to another, all in a continuous flow. Before you begin the process, therefore, you should make sure there is plenty of room left on your disk for the extra file that will be created.

The recommended sequence is as follows.

If you have material in the computer's memory which you wish to preserve, `SAVE` it now.

Check the files on your disk. Type

`*.` **RETURN** or `*CAT` **RETURN**

Are there any files you want to delete? If so type

`*DELETE (filename)` **RETURN**

Compact your disk. This rearranges the files as economically as possible. Type

```
*COMPACT RETURN
```

Warning: Compacting destroys any text currently in the computer's memory.

You are now ready to start the **EDIT** procedure, by reading in your file and giving a name to the file to which the text you are processing will go. Type

```
EDIT (file-in) (file-out) RETURN
```

(In our example, 'file-in' and 'file-out' represent the names of the files.)

The system reads text in from your file ('file-in'). You process it. When you are ready to go on to the next part, you type

```
MORE RETURN
```

The text you have processed is written into your 'file-out' and new text is read in from your 'file-in'.

When you wish to stop editing, even if you have not finished the document, type

```
FINISH RETURN
```

and the text in memory is put into your 'file-out'. Any unread text left in 'file-in' is read and transferred to 'file-out'.

You are now left with two files, the original file ('file-in') and the revised one ('file-out'). If you no longer need the original type

```
*DELETE (file-in) RETURN
```

The command **QUIT** is an alternative to **FINISH**, which simply abandons the editing, leaving the 'file-out' incomplete. You should delete 'file-out'.

Using cassette tape

If your computer is set up for disk, and you wish to use tape, type

```
*TAPE RETURN
```

To record files

Make sure there is a blank tape in the recorder.

Type

```
SAVE (filename) RETURN
```

The 'cassette motor' lamp comes on. The message appears on screen:

```
RECORD THEN RETURN
```

Press RECORD on your cassette recorder and **RETURN** on the computer. When the prompt (=>) returns the file is recorded.

If your cassette recorder has no motor control (REM) socket, stop it quickly. (If it has a REM connection, it will stop automatically.)

To record part of a file

Set markers 1 and 2 at the beginning and end of the part you wish to record. Use the procedure above, except that at stage 2 type

```
SAVE (filename) 1 2 RETURN
```

To read files

Wind back the cassette to the appropriate point.

Type

```
READ (filename) RETURN
```

The 'cassette motor' lamp comes on.

Press PLAY on the cassette recorder. Depending on the cassette recorder, you may hear high or low pitched sounds. When the prompt (=>) returns the file has been read in. This procedure deletes any text currently in memory and replaces it with the new file.

`WRITE` has the same effect as `SAVE`.

`READ` does not delete text currently in the computer's memory, but adds to it, in so far as memory allows. To replace a file in memory, use `NEW` (see page 23Ø) before you `READ`.

To read a file into a specific place in existing text

Set marker 1 at the place where you want the file read in. Use the procedure above, except that at stage 2 type

`READ (filename)` **RETURN**

Note: File names can have up to ten characters, and must start with a letter. *They must not include spaces or punctuation marks.* Refer to Into VIEW for detailed instructions of loading and saving with cassette tape.

Word counting and formatting

`COUNT` **RETURN** gives the number of words in the text in memory. To count the number of words in a part of the text, set markers 1 and 2 at the beginning and end of that part. Then type

`COUNT 1 2` **RETURN**

Warning: If you are using widely spaced titles, you will get misleading results when you ask for a word count. VIEW counts the spaces rather than the words, and the spaces are counted as a series of single-letter words.

Note: We have suggested using markers 1 and 2 because they are visible, but `COUNT` will work with any markers.

`FORMAT` is a global formatting command, allowing you to format all text in memory, or a selected part of the text. Its effect is the same as that of the FORMAT BLOCK key, but it formats the entire file rather than single paragraphs.

To format all text in memory, type

FORMAT **RETURN**

To format a selected part of the text, first set markers 1 and 2 before and after the part you wish to format. Then type

FORMAT 1 2 **RETURN**

CLEAR **RETURN** clears markers 1 and 2.

Finding words in the document

VIEW allows you to search the whole document or a specified part of the document for any word you select. The system then finds the word and indicates it.

SEARCH

To search the whole document for a word type

SEARCH (word) **RETURN** or S (word) **RETURN**

If the word is in the document, the screen will switch to text mode and you will find the cursor under the first letter of the first occurrence of that word. To move on to the next occurrence of the word press NEXT MATCH on the red function keys.

If the word is not present in the document VIEW will remain in command mode.

Limited search

To search part of the document, set markers 1 and 2 before and after the part you wish to search. Then type

SEARCH (word) 1 2 **RETURN**

Wild search

If you are not sure whether the word you are seeking has been spelled correctly, you can replace some characters with a question mark. For example

```
SEARCH THE?? RETURN
```

will find both THERE and THEIR since the question mark stands for any character. You can also specify **RETURN** and **TAB** in your search. Vertical bar (|) is **RETURN**; tilde (~) is **TAB** . If you want to replace the question mark with another character, use

```
WILD (character) RETURN
```

To change a word every time it occurs

Type

```
CHANGE (old word) (new word) RETURN  or
C (old word) (new word) RETURN
```

The system will remain in command mode, but the word will be changed throughout the document.

To limit the effect of the changes to part of the document, first set markers 1 and 2 before and after the part of the document concerned. Then type

```
CHANGE (old word) (new word) 1 2 RETURN
```

Replacing a word at choice

Type

```
REPLACE (old word) (new word) ((return))  or
R (old word) (new word) RETURN
```

The display switches to text mode.

The letters RP appear in reversed type at top left of the screen. The cursor is at the first character of the first occurrence of the word. If you do not want that particular word replaced press N. If you want it replaced press Y. The word will change; the text around it will be adjusted; and the cursor will go to the next occurrence of the word. Press **ESCAPE** to return to command mode.

Upper and lower case

With the CHANGE and REPLACE commands VIEW normally copies the case of the characters in a word – thus if you had the word 'Washington' in the text and issued the command CHANGE washington boston the result would be 'Boston'.
This facility is known as 'folding'.

If you want to change the case of some of the letters in a word (whether or not you are changing their spelling) you will have to turn the folding off. To do this before using CHANGE or REPLACE type

FOLD 0 **RETURN**

You could then for example replace 'aspirin' with 'Aspirin'.
To turn folding on again type

FOLD 1 **RETURN**

To find out whether folding is on or off, type

FOLD **RETURN**

Folding is on by default. In other words, CHANGE will search for the string you specify disregarding whether it is upper or lower case. If you turn folding off with the command

FOLD 0 **RETURN**

it will search for the target string with the case as specified by you, and replace it with the replacement string in the case specified by you.

If you want to change two or more words that occur together throughout the text you must use delimiters to enclose them. For example

```
CHANGE/Washington D C/New York/
```

In this case the delimiter is the slash. The first character after CHANGE indicates the delimiter.

Printing

VIEW operates most printers, serial or parallel. If a printer works with the BBC Microcomputer, it should work with VIEW too.

For detailed instructions on how to use a printer, see the Printer Driver booklet.

General procedure

Insert the printer driver disk (or tape).

Switch to command mode and type

```
PRINTER (name of printer) RETURN
```

This loads a program which operates the printer. The words `Printer (name)` appear on the screen.

If you require microspacing, type

```
MICROSPACE RETURN
```

The letter `(m)` appears after the name of the printer.

(If your printer driver cannot handle microspacing, VIEW tells you so.)

Printing the text you have composed can be done either from the text you see on the screen (ie the text in the edit memory), or by inputting a file from disk or tape and outputting it directly to the printer. Printing a file from disk or tape does not in any way affect the text currently in memory, so you may hold one block of text in memory for editing, while printing another.

PRINT

This command causes the whole of text in memory, or the whole of the file named in the command to be printed. It is therefore only suitable if you are using continuous stationery or if the text is very short.

To print the text in memory type

`PRINT` **RETURN** or `P` **RETURN**

To print a file type

```
PRINT (filename) RETURN  or
P (filename) RETURN
```

For example:

```
PRINT agenda RETURN
```

To print several files type their names with a space between each. For example:

```
PRINT agenda report letter RETURN
```

SHEETS

This command outputs a page at a time to the printer.

To print the text in memory type

```
SHEETS RETURN
```

To print a file type

```
SHEETS (filename) RETURN
```

The following message appears on the screen:

```
Page 1..
```

To print, press any key except M, Q, **ESCAPE** or **COPY**.

To miss out a page, press M.

To quit printing, press Q.

To print several files a page at a time, type their names separated by spaces. For example:

```
SHEETS agenda report letter RETURN
```

Screen

This is not strictly a printing command. It takes material from disk or tape or from the edit memory, and outputs it to the screen. This is useful for making a rough check on the layout of the text, although the printer drivers will not be active and variations in line length may distort its appearance. Most useful for checking where page breaks occur.

To display text in memory type

```
SCREEN RETURN
```

To display a file type

```
SCREEN (filename) RETURN
```

As with the **PRINT** and **SHEETS** command it is possible to display a series of files. For example:

```
SCREEN agenda report letter RETURN
```

The **SCREEN** command displays the text a screenful at a time. To move on to the next screenful, press **SHIFT**.

Amendments to Disk Filing System User Guide

Chapter 2 Getting going Page 7 onwards

The BBC Microcomputer has a cable clamp plate attached to its underside which is held in place by six nuts (see Fig 5 in the BBC Microcomputer User Guide Part 1). Before the disk drive can be connected to the BBC Microcomputer this cable clamp plate must be removed by undoing the six nuts. Plug the ribbon cable into the socket on the BBC Microcomputer and then replace the cable clamp plate and tighten the six nuts just enough to clamp the ribbon cable gently.

If your disk drive has its own power supply then the only connection between the drive and the BBC Microcomputer is through the ribbon cable (see Fig 1).

Fig 1 does not show the cable clamp plate. See Fig 5 in the BBC Microcomputer User Guide Part 1 for more information.

Chapter 3 Disks Page 1Ø

Paragraph 2 should read:

Two types of disk drive are available for the BBC Microcomputer, a single unit (4ØØKbytes) and a dual unit (8ØØKbytes). Each disk will store about 4ØØ,ØØØ characters, equivalent to 2ØØ pages of this book. All disks used must be double sided double density soft-sectored, and formatted 8Ø track by the user using the `*FORM8Ø` utility supplied on the introductory and utilities disk.

Page 13

The section headed 'Proceed as follows' should read:

If it has not been done already write-protect the utilities disk with a tab, as described. This is very important otherwise you might lose all the information on the disk.

Insert the utilities disk into drive Ø and type

`*FORM8Ø` **RETURN**

Close the drive door and you should see the following message

```
Disk formatter 1.10
Format which drive ?
```

Remove the utilities disk and put a new disk in its place. Do not forget to change the disk over. If you format the utilities disk you will lose everything on it.

In reply to the question displayed type

```
0
```

The computer will say

```
Do you really want to format drive 0 ?
```

If you are sure that the disk in drive 0 is a new one type

```
Y
```

and close the drive door.

What the computer now does is called 'formatting' which prepares the new disk for use with the BBC Microcomputer. A series of numbers is displayed, and when the formatting is finished the computer says

```
Disk formatted - repeat (Y/N) ?
```

The disks are double sided and so both sides must be formatted. The other side of the disk in drive 0 is in drive 2, so type

```
Y
```

followed by

```
2
```

followed by

```
Y
```

and the computer will format side 2. When this is done you can remove the new disk and put the utilities disk back in again.

If you have a dual drive, put the new disk which you have just formatted into drive 1 (the bottom drive) and type

```
*COPY01*.* RETURN
```

The computer will now make a complete copy of the introductory and utilities disk on the new disk.

If you have a single drive, type

```
*COPY00*.* RETURN
```

then follow the instructions on the screen to insert the utilities (source) disk and the new (destination) disk alternately (see Fig 4).

If the message

```
Disk read only
```

appears, it means you had the wrong disk in the drive. Start again by pressing **ESCAPE**.

Amendments to Into VIEW and VIEW Guide

Two important general notes

1. When performing a Disk Filing System operation in VIEW, the **ESCAPE** key must not be used to abort the operation. All text may be lost if this is done.

For example, if there is a disk in a drive with the drive door open and you type

```
SAVE  filename
```

then you must not abort by pressing **ESCAPE**. The `SAVE` operation must be completed.

2. The start-up mode on the BBC Microcomputer is MODE 4 and not MODE 7 as stated.