



**BBC MICROCOMPUTER**

**DFS MANUAL**

## CONTENTS

1	What is a disc system? A disc drive Disc filing system Controlling the filing system Summary	3
2	Getting Going Connecting the disc drive Starting the filing system Copying the master disc	7
3	Discs Handling Prevention of accidental erasure Tracks, sectors and bytes What is formatting?	9
4	Disc files File specification Multi-file operations Auto-start facilities Library files	16
5	The filing system commands	20
6	The filing system utilities Formatting	50
7	Random access files	53
8	Using the filing system in assembler General principles Read/write one byte Read/write a group of bytes Read/write a sector	57
9	Changing filing systems	63
10	Error messages	64

11	Technical information	66
	18 bit addressing	
	Disc catalogue	
	File system initialisation and !BOOT	
	The floppy disc drive parameters	
12	Filing system command summary	69
	Index	71

# 1 WHAT IS A DISC SYSTEM?

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

## A DISC 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 re-type 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 computer 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:

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

A disc system does not have these disadvantages.

To help you to understand how a disc system works, we shall draw some comparisons with a filing cabinet. A disc system always includes at least one disc drive. The BBC computer's disc drives are buff coloured metal boxes. The front of a disc drive is black. These are floppy disc drives, as distinct from fixed disc drives, which you may have heard of. The disc drive can be compared to an empty filing cabinet with no drawers in it yet.

Just as a filing cabinet is pretty useless without drawers, so a disc drive cannot do much without discs. The discs used with the BBC computer are 5.25" soft-sectored floppy discs. They can be inserted and removed from the disc 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. Diagram 1 on page 7 shows the correct way to insert a disc. To reinforce this:

Discs 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 disc go in first.

**NB** The back of the disc drive has some cables coming out of it. These are to connect the computer.

Diagram 2 shows and names the various parts of a disc and chapter 3 "Discs" gives more detailed information about them. The discs hold the information. You can change the discs in the drive just like you can change the cassettes in a cassette recorder. So you can use lots of discs to store information, but you can only read the information from one disc at a time. Just as a music cassette may have several different songs on

it, a disc may have different groups of information and these 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 comparisons with a filing cabinet, opening a drawer and throwing all the papers in 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 disc. When you first buy discs, they are blank - like empty drawers. Before the computer can put any information on them, the discs are first prepared by having marks put on them, which divide the discs into sectors. "Sectors" is the name given to a set of equal divisions created on the disc by the computer. (See diagram 5). This operation is called "Formatting" and is fully described in chapter 6.

When you insert the disc into the disc drive and close the drive door a rotating boss engages with the central hole in the disc and spins the magnetic disc inside its protective jacket. (Do not confuse the protecting jacket with the disc envelope, see diagram 2). In order to read or write information on to the disc the disc drive has a "Read/Write head". This head is designed to move in and out along the "Head Slot" in the disc jacket. This head actually rests on the surface of the magnetic disc as it rotates inside the jacket. When you want to read some of the information on the disc, 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 disc 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".

#### **DISC FILING SYSTEM**

We mentioned five comparative disadvantages of using a cassette recorder to store information. These may be partly summarised by saying:

You have to control the cassette recorder and keep track of the information on it.

When using a disc this is all done for you by the "Disc Filing System".

The disc filing system is a machine code program produced by the computer manufacturer. On the BBC computer 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 disc drives are controlled by the computer using this program. When you prepare new discs by formatting them this is done

by the disc filing system. When you **SAVE** one of your **BASIC** programs the disc filing system does the following:

- Starts the disc drive working
- Finds a free place on the disc big enough for your program
- Makes a note of where it put your program in order to be able to find it again.
- Moves the disc 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 disc
- Stops the disc 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 disc system as well as the cassette system. However, the disc filing system puts the name to special use. The first two sectors on every disc are reserved for a "catalogue" when the disc is formatted.

The name of your program, referred to as a "filename" is written into the catalogue together with the number of the sector on the disc 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 catalogue to find out where on the disc to find that file, and then moves the read/write head to that exact place on the disc. The file is then loaded into the computer's memory (**RAM**) automatically. This illustrates another advantage of a disc drive. The read/write head can be quickly moved to any point on the disc with great accuracy. (Incidentally, the precision engineering needed to accomplish this explains why disc 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 disc catalogue, 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. It is not hard to see why microcomputers have become so popular in offices.

## CONTROLLING THE FILING SYSTEM

The filing system controls the disc drive but in turn we must be able to give instructions to the filing system. Two ways are provided. One is by typing a special command word preceded by the "\*". These are all listed in chapter 5 together with details of their functions. Any of these direct commands can be incorporated in a program if required. Chapter 7 of this manual describes the use of a number of BASIC keywords, with special reference to files created on a disc. All these keywords are introduced in the main User Guide.

### Summary

- A disc system includes a disc drive, some discs, a connection to the computer and a machine code program permanently in the computer called a "disc filing system".
- A disc is inserted into the disc drive where it spins round inside its protective jacket.
- The disc drive's read/write head moves in and out along a radius of the disc as it spins around.
- The disc filing system controls the disc drive and the movement of the head.
- Discs are divided into sectors by the filing system and the first two sectors are reserved for a catalogue.
- Programs and other information are stored on the disc and are given a "filename".
- By reference to the catalogue the filing system can find any information on a disc associated with a specified filename.
- 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 disc drive to the underside of the computer as shown in diagram 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. However where this aid is not present 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:

```
BBC Computer
Acorn DFS
BASIC
>
```

This indicates that the disc filing system is installed and working O.K. Now press **SHIFT** and **BREAK** holding both keys down together. The disc 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 disc is properly connected and working. If nothing happens, check the connections between the disc and computer.

The foregoing assumes that the auto-start option has been set to work when **SHIFT** is held down with **BREAK** - see Chapter 10.

Insert the utilities disc into drive 0. (With dual drives, drive 0 is the one directly connected to the computer. See diagram 1, page 7). Press **SHIFT** and **BREAK** again. This time the following message will appear on the screen:

```
Midwich Computer Co Ltd
Utilities Disc
Press F0 for 40 Track, press F1 for 80 Track
```

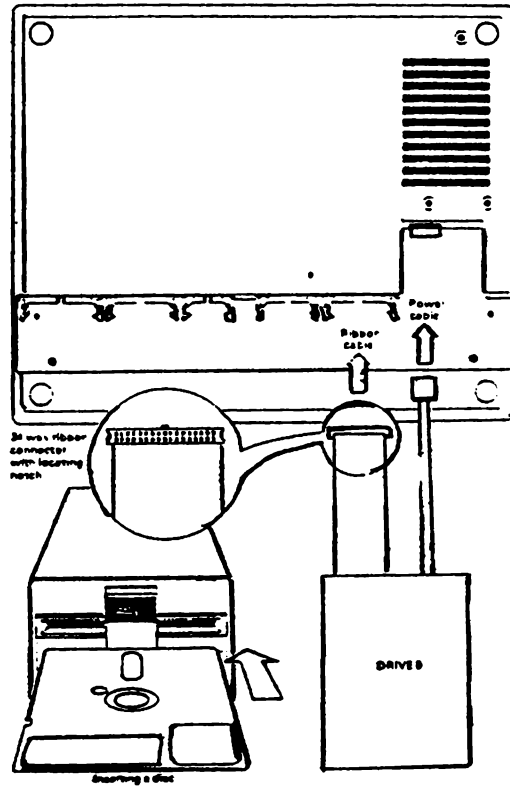
Pressing **F0** or **F1** produces the following display:

```
F0 Format Drive 0
F1 Format Drive 1
F2 Format Drive 2
F3 Format Drive 3
F4 Verify Drive 0
F5 Verify Drive 1
F6 Verify Drive 2
F7 Verify Drive 3
```

This is an example of a **!BOOT** file which will be explained in more detail later. When the **>** prompt reappears on the screen you can start entering your programs of filing system commands, but before you do that, read the next two chapters: "Discs" and "Disc Files".



**Diagram 1 - Connecting the disc drive**



### 3 DISCS

The BBC Microcomputer uses 5 25" discs for storing information. You may have heard them referred to as a "floppy discs", "discettes", or "mini-discs", we will always refer to them simply as discs. (The American spelling is disk).

#### HANDLING

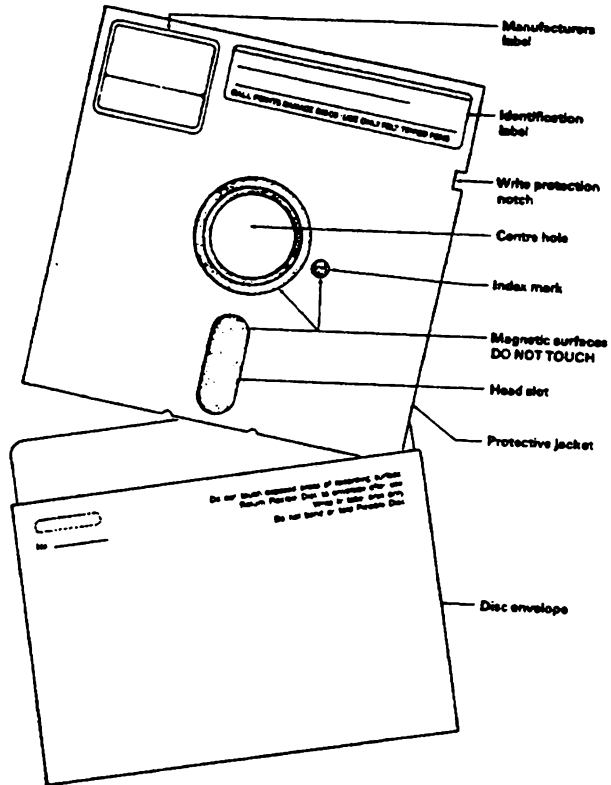
Discs should be handled with care to avoid physical damage or damage to the recorded information. Diagram 2 will help you to identify the various parts of the disc that we are referring to. The following guidelines should be observed:

- Do not try to remove the circular magnetic disc from the square, black protective jacket covering it.
- Do not touch the exposed recording surfaces.
- Avoid dust. Put the discs back into their envelopes when they are not in the disc 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 discs into the drive carefully. If it rotates noisily open the drive door and adjust it.

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

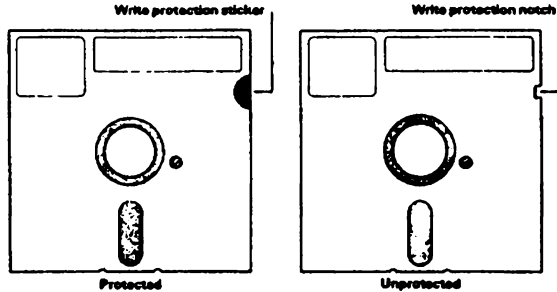
The foregoing is deliberately comprehensive but do not let it frighten you from using the discs. Handled sensibly, a disc will give good service.

**Diagram 2 - A 5.25" disc**



## PREVENTING ACCIDENTAL ERASURE

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



Another way of protecting important information is to keep several copies of it on different discs. 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. Applying this principle to protecting your own information will be beneficial.

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 discs, each with a copy of the information from the mastered disc on it. On day 4 the master would be copied to the grandfather again and so the cycle continues. In business where information on discs changes from day to day this regular routine is important. For personal computing it is not 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 making copying information easy. These are \*BACKUP and \*COPY which allow you to copy a complete disc or specified sections of it. See diagrams 3 and 4 and the appropriate sections of chapter 5 for full details. We suggest that you make a copy of your utilities disc now!

### Proceed as follows

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

Insert the utilities disc into drive 0 and type **SHIFT** and **BREAK** select **F0** if you have 40 Track Drives or **F1** if you have 80 Track Drives.

Now press the function key corresponding to the task you wish to carry out.

When formatting, the machine will respond with the following message.

nn Track Formatter Version 1.00

Format drive \*\* (nn tracks) Y/N :

Where nn is either 40 or 80 depending on the type of drive you are using and \*\* is the drive No. you have chosen.

When verifying, the machine will respond with the following message.

Disc Verifier Version 1.00

Verify drive \*\* Y/N :

Where \*\* is the drive No. you have chosen

IN EITHER CASE PLACE THE DISC YOU WISH TO FORMAT OR VERIFY IN THE DRIVE CHOSEN. THEN ANSWER Y TO THE PROMPT. Any other answer will result in the program stopping and you will have to start again.

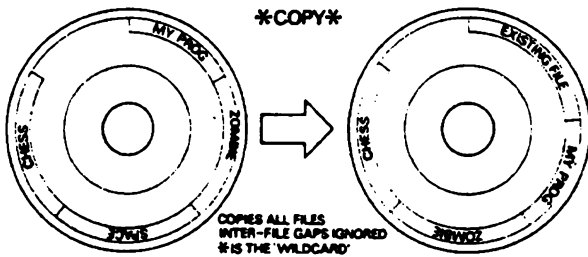
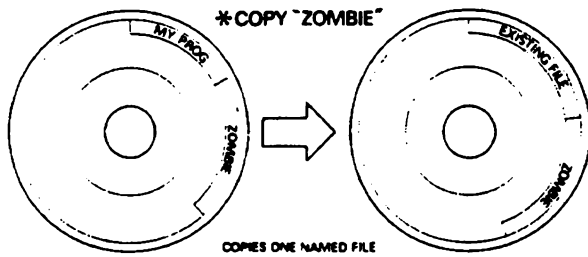
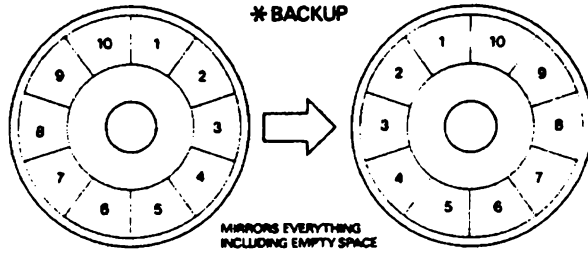
#### NOTES.

Formatting prepares new discs for use by the computer. (See page 14)  
FORMATTING DESTROYS ANY DATA ALREADY ON THE DISC.

Verifying is a process of quickly reading the whole of the disc to check that it is okay.

When you have formatted your first new disc it may be a good idea to make a copy of your utilities disc using the \*BACKUP utility (page 22)

Diagram 3 - Copying



## **TRACKS, SECTORS AND BYTES**

Information is written on to the disc in concentric circles, called tracks. Each track is divided into ten sectors. Each sector is further divided into 256 bytes. Space on the disc and in the computer's memory is measured in bytes. One byte corresponds to one character. 256 of the bytes in each sector are available for storing your programs and data. From this it follows that a 40 track single-sided, single track density disc of the type used in the single drive unit will hold:

40 tracks x 10 sectors x 256 bytes = 102400 bytes

or characters of information. The dual drive unit uses double-sided double track density discs. Therefore with 80 tracks on each side, one disc will hold

160 tracks x 10 sectors x 256 bytes = 409600 bytes

or characters of information.

## **WHAT IS FORMATTING?**

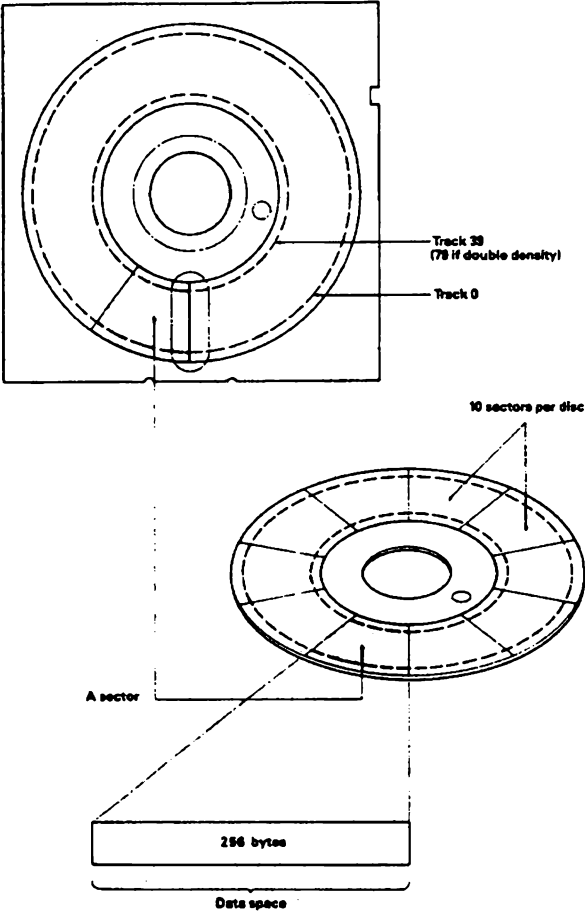
The disc filing system automatically records the location of your programs and data on the disc. The first two sectors on track zero of a disc are reserved for this purpose. The "catalogue" of a disc is recorded in these sectors. Whenever you wish to access a piece of information on the disc, the filing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

Clearly before a disc can be used by the filing system for storing your information it must have these "reference marks" put on it. All new discs must be prepared in this way. The process is called "formatting". It includes setting up the track and sector format on the disc and creating the catalogue. (Full details are provided in the technical information on page 66).

Each sector on each track is given a unique three digit identifier in the catalogue. The first two digits are the track number, the last one is the sector number. The sectors are numbered 0 to 9 and the tracks 00 to 39 or 79 if it is an 80 track disc.

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

Diagram 5 - Tracks, sectors, bytes





## 4 DISC FILES

Probably the first thing you will want to do with the filing system is to record one of your programs onto a disc. You can do this simply by using the **SAVE** command in BASIC and the filing system takes care of the rest. (NB 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 on to the disc. When **SAVE'd** the program must be given a name. This is called the filename and is used to refer to the program if you later want to copy it back from the disc. Each program **SAVE'd** on to the same drive must be given a unique name. The format of the **SAVE** command is:

**SAVE "filename"**

where "filename" can be up to 7 characters. Letters and digits are allowed. The characters

# \* . :

have special meanings which are explained later.

The filename is written into the catalogue together with the sector number where the information starts. Next time you refer to the filename the filing system checks the catalogue to see where the information has been placed on the disc, 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. Filename  
:<drv>. <dir>. <Filename>.  
e.g:  
: 1. 2. MYPROG1

Notice the drive number, directory, and filename are separated by full stops. 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 0 to 3 and preceded by a : (colon). The colon in effect 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 disc is given a separate drive number.

Single-drive, single-sided  
Drive 0  
-----

Dual-drive, double sided

Drive 0	Drive 1
-----	-----
Drive 2	Drive 3

The effect of including the drive number in the full specification is that :1\$.MYPROG 1 is different from :2\$.MYPROG1 although the file names are the same, they are on different drives.

## DIRECTORIES

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

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

because the directory is different.

## FILENAMES

The filename can be up to seven of most of the characters on the keyboard in any combination, except the four previously mentioned. 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 0 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

:0\$.MYPROG

assuming you have not changed the current drive and directory. (Chapter 5 "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 "Wildcard" 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, e.g:

\*INFO :0\$.MYPROG

will display information about the file named MYPROG in directory \$ on drive 0.

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

**\*INFO :0\$.MYPROG**

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

**\*INFO :0\$.MYPROG#**

means; Display information about all files on drive 0 on directory \$ with names starting "MYPROG" followed by any SINGLE character. e.g:

MYPROA, MYPROT and MYPROG and so on.

The character \* means multiple #'s to the end of the field, e.g

**\*INFO :0\$.M\***

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

#### **AUTO-START FACILITIES**

Sometimes it is useful to make a program or a file on one of your discs \*LOAD, \*RUN or \*EXEC automatically when you insert the disc and press BREAK. This can be done using a file named !BOOT.

!BOOT is a special filename recognised 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 and the disc using \*OPT 4,n see chapter 5.

Option 0: ignores !BOOT

Option 1: \*LOAD's !BOOT into memory

Option 2: \*RUN's !BOOT as a machine code program not a BASIC program

Option 3: \*EXEC's !BOOT

See Chapter 5 "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 filing system utilities disc is loaded using a !BOOT file.

As well as programs, you may wish to store data on the discs. The filing system provides special 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 7.

## **LIBRARY DIRECTORY**

The disc file 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)**

it is equivalent to typing

**\*ROW (Utility name)**

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 FILING SYSTEM COMMANDS

The Disc Filing System is an 8K byte program. BASIC programs are stored on a disc or tape, but the filing system is stored in Read Only Memory (ROM) inside the BBC computer. The filing system controls the reading and writing of information to and from the discs 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 recognise 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

### PURPOSE

A plain English description of what the command does.

### EXAMPLES

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

If appropriate 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 a 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 disc causes the one on the disc 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 un-sets 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 **File locked** is produced.

If you attempt to use **\*ACCESS** on a write protected disc the message **Disc write protected** is produced.

### **IMPORTANT**

Locking a file does NOT prevent it from being removed from a disc with **\*FORM40** or **\*FORM80** or from being overwritten with **\*BACKUP**.

## **\*BACKUP (SOURCE DRV) (DEST. DRV)**

### **PURPOSE**

To read all the information on one disc and write it to another, producing two discs 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 alternatively ask you to insert the source and destination discs into drive 0. This makes it possible to copy discs even if you only have a single drive. A 40 track disc is copied in 5 sections.

Diagram 4 in chapter 3 illustrates the process.

All the information previously on the destination disc is overwritten so be careful not to confuse the source and destination discs. If the source disc is blank the destination disc 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 IBOOT file described in Chapter 3.

### **EXAMPLE**

**\*BUILD IBOOT**

will cause everything subsequently typed in to be entered into a file called IBOOT

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

```
>*BUILD IBOOT
00010 FIRST LINE OF TEXT
00020 SECOND LINE
00030 ESC
```

Typing ESC on a line by itself terminates a \*BUILD command.

### **DESCRIPTION**

Builds a file from the keyboard.

### **ASSOCIATED COMMANDS**

**\*EXEC**

**\*LIST**

**\*TYPE**



## **\*CAT <DRV> CATALOGUE**

### **PURPOSE**

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

**EXAMPLE:**

<b>*CAT 0</b>	
<b>PROGRAM (nn)</b>	
<b>DRIVE: 0</b>	<b>Option:2 (ROM)</b>
<b>Directory: 0.\$</b>	<b>Library :0.\$</b>
<b>!BOOT</b>	<b>HELLO</b>
<b>SUMS</b>	<b>TABLE</b>
<b>TEST</b>	<b>VECTORS</b>
<b>ZOMBIE</b>	
<b>A.HELLO L</b>	<b>B.SUMS</b>

Note that the heading part of the catalogue shows the drive number, the title of the disc, the currently set auto-start option of the disc (in this case 2 for ROM), 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 disc. !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 disc catalogue.

### **ASSOCIATED COMMANDS**

**\*INFO**  
**\*ACCESS**  
**\*TITLE**  
**\*OPT4,n**  
**\*DIR**  
**\*DRIVE**

### **NOTES**

Permitted values of <drv> are 0, 1, 2 or 3 nothing. Other values will cause the message

**Bad drive**

to be displayed and you will have to re-enter the command correctly. The top two lines of the catalogue include the disc title, disc option, drive number and current directory.

### **IMPORTANT**

The Catalogue, hence the disc, will hold up to a maximum of 31 files.

## **\*COMPACT <DRV>**

### **PURPOSE**

Attempting to **SAVE** a program or file on to a disc may produce the message "Disc Full" if there is no single space available on the disc 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 disc to the end. When you delete a number of files, the spaces they occupied will probably be distributed over the disc with current files in between them. **\*COMPACT** moves all current files to the "start" of the disc leaving the spare space in one continuous block at the end.

### **EXAMPLE**

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

As "compacting" proceeds all the current files are displayed in the order in which they occur on the disc.

### **DESCRIPTION**

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

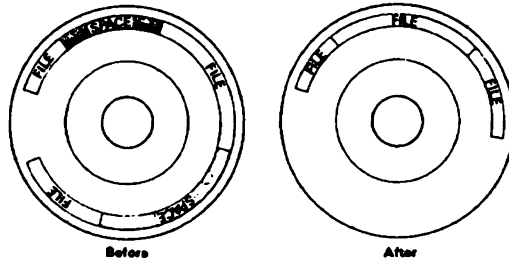
### **ASSOCIATED COMMANDS**

**\*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.

Diagram 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 disc 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 "wildcard" facilities may be used to specify a group of files to be copied e.g:

**\*COPY 0 1 \*.MY\***

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

**DIAGRAM**

Diagrams 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 catalogue of a disc. The space occupied by the file becomes available for other information. Succeeding file names in the catalogue 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 disc.

### **DESCRIPTION**

Single file deletion.

### **ASSOCIATED COMMANDS**

**\*WIPE**

**\*DESTROY**

**\*COMPACT**

### **NOTES**

If the disc is write-protected the message

**Disc write protected**

is produced.

If the file is not found in the directory the message

**File not found**

is displayed. If the file is locked the message

**File locked**

appears.

Once deleted a file cannot be restored.

## **\*DESTROY <AFSP>**

### **PURPOSE**

To remove specified files from the disc in a single action. This command takes 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>). The current directory is always set to "\$" when you press **BREAK**. To save files in a different directory in the catalogue you must use this command to change the current directory to the one you want and then save them.

### **EXAMPLE**

#### **\*DIRA**

This sets the current directory to A. You now have to access to any files in directory A in the catalogue. 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 catalogue. It merely states which directory in the catalogue you have access to by default.

## **\*DRIVE <DR> SET CURRENT DRIVE**

### **PURPOSE**

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

### **EXAMPLE**

**\*DRIVE 1**

sets the current drive to 1 and

**\*CAT**

will produce a catalogue of drive 1

**\*CAT 0**

will catalogue drive 0 but the current drive is still drive 1 until you change it back to 0 or press **BREAK**

### **DESCRIPTION**

Sets the current drive



## **\*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 3 and in the section on OPT4 in this chapter. If you create a !BOOT file containing the BASIC keyword CHAIN followed by the filename of one of your BASIC programs, the effect of pressing SHIFT BREAK will be to automatically load and run the BASIC program.

## **\*HELP <KEYWORD>**

### **PURPOSE**

Displays useful information on the screen. In the disc 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 disc filing system are **UTILS** and **DFS**

### **EXAMPLE**

**\*HELP DFS**

**DFS 1.00**

**ACCESS** <asfp> (L)  
**BACKUP** <source drv><dest><drv>  
**COMPACT** (<drv>)  
**COPY** <source drv><dest drv><afsp>  
**DELETE** <fsp>  
**DESTROY** <afsp>  
**ENABLE**  
**INFO** <afsp>  
**RENAME** <old fsp><new fsp>  
**DIR** <dir>  
**DRIVE** <drv>  
**LIB** <dir>  
**TITLE** <disc name>  
**VERIFY** (<drv>)  
**WIFE** <afsp>

**\*HELP UTILS**

**DFS 1.00**

**BUILD** <fsp>  
**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 disc filing system. **\*HELP** is a Machine Operating System command.

## **\*INFO <ASFP>**

### **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	Filename	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

**File not found**

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

The wildcard 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=0, A=1 and X=the address of the 4 byte block in page 0 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 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 8 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 8 provides more information about using the disc 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 tokenised so listing a BASIC program file will display nonsense. (See 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 screen full until you press either **SHIFT** key to make it continue. **CTRL W** turns page mode on, **CTRL O** turns it off.



## **\*LOAD <FSP> <ADDRESS>**

### **PURPOSE**

Reads a named file from the disc into the 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 1900 (hex), which is the load address of the file when it was saved.

(see example in \*INFO)

#### **\*LOAD HELLO 3200**

Reads the file HELLO into memory starting at location 3200 (hex).

Other examples are

```
LOAD    "HELLO"  
LOAD    "HELLO" 3200
```

### **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 disc. If the file is not found the message

File 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 disc is accessed the information is displayed. (n) can be anything from 1 to 99 to enable the feature. (n) =0 disables it.

### **EXAMPLE**

**\*OPT 1 1** or **\*OPT 1, 1**

enables the messages;

**\*OPT 1 0** or **\*OPT 1, 0**

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 disc in the currently selected drive. There are four options to choose from 0,1,2 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 0.

### **EXAMPLE**

\*opt 4 0 does nothing  
\*OPT 4 1 will \*LOAD the file !BOOT  
\*OPT 4 2 will \*RUN the file !BOOT  
\*OPT 4 3 will \*EXEC the file !BOOT

### **DESCRIPTION**

Changes the start-up option of a disc.

### **NOTES**

It is essential to include a space between the command and (n).  
\*OPT40 would produce the message.

#### **Bad option**

If the disc is write-protected the error message

#### **Disc write-protected**

is produced in response to the \*OPT 4 command.

If the option 0 is set the !BOOT file need not be there. With any other option the message

#### **File not found**

is produced if !BOOT is not found in directory \$ on drive 0.

### **IMPORTANT**

Do not confuse \*OPT 4 with BASIC keyword OPT or \*OPT 1. They are completely different. Refer also to Chapter 11 where it describes how to swap the effects of **BREAK** and **SHIFT-BREAK**.

The utilities disc 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**

This is not allowed. The file cannot be moved from drive 0 to drive 1 using **\*RENAME**. Only the directory and filename can be changed.

If the file does not exist the message

**File not found**

is displayed. If the first file is locked

**file locked**

is displayed. If the disc is write-protected

**Disc write-protected**

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

**file 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 ADDRESS>) (<RELOAD ADDRESS>)**

#### **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 disc 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 "PROC" SSSS FFFF EEEE RRRR**  
**\*SAVE "PROC" SSSS +LLLL EEEE**

**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 disc is write-protected the message

**Disc write-protected**

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

**Directory full**

is produced. If the specified filename already exists and is locked the message

**File locked**

is produced.

If the file already exists but is unlocked it is deleted. Then starting in sector 2 track 0 a gap large enough to hold new information is searched for. If none is found the message

**Disc full**

is produced.

If enough space is available, the information is written on to the disc and the filename is entered on to the catalogue in the current directory.

## **\*SPOOL <Fsp>**

### **PURPOSE**

Prepares a file of the specified name on the disc 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 disc into memory.

**\*SPOOL TEXT**

opens a file called **TEXT** on the disc 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 **TEXT**

**\*SPOOL**

turns off the "spooling" and closes the file called **TEXT**

### **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 "tokenised". This means that the lines which you type in your program are abbreviated inside the computer's memory and on the disc. A program file will contain these abbreviated "tokens" rather than your original program text.

Displaying the file using **\*LIST** will therefore produce strange results. The example above shows you how to create a file containing your original program text. If you display that file using **\*LIST** your program will appear just as you typed it in.

## **\*TITLE <DISC NAME>**

### **PURPOSE**

Changes the titles of the disc in the current drive to the first twelve characters after the command. It fills in with "nulls" if there are less than twelve characters. Any characters are allowed.

### **EXAMPLE**

**\*TITLE "MY DISC"** with five "nulls" 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 disc is write-protected the message

**Disc write-protected**

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 disc as text files when you **SAVE** them so this command will display nonsense.

Page mode is selected with **CTRL N** and turned off by **CTRL O**

## **\*WIPE <AFSP>**

### **PURPOSE**

Removes specified files from the catalogue and rearranges the catalogue. 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 filename 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).

## 6 THE FILING SYSTEM UTILITIES

As explained in Chapter 5 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 utilities disc. They are used in the same way as the commands by typing

\* (Utility name)

The utilities disc must be in the disc 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 are two "formatting" utilities supplied; one for 40 track discs; the other for 80 track discs. On the utilities these are normally in drive 0 directory \$, which are the default values on the start-up for both the Library and the Current drive/directory. (See command \*DRIVE and \*DIR). The third utility is \*VERIFY.

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

**\*FORM40 (<drv>) Format a 40 track disc**

### **Purpose**

The command prepares new discs for using with the filing system on the BBC microcomputer. It marks areas of a disc where information will be stored and sets up a catalogue. The catalogue is empty at first but when you store programs and data on the disc the catalogue records their position on the disc. They can then be retrieved quickly by reference to the catalogue. While formatting, the information put on to disc is verified automatically.

### **Example**

- 1) To format a new disc, insert the utilities disc into the disc drive and type

**\*FORM40**

The computer asks

Do you really want to format drive 0?

Remove the utilities disc insert the disc 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
disc formatted
```

Above is the response if the formatting was successful.  
If not, the formatting will stop and 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 disc straight away.

#### Description

Initialises discs with track and sector format. Clears the catalogue and verifies the sectoring.

#### Associated commands

\*FORM80

#### Notes

\*FORM80 is the same except that it formats the 80 track discs used in the dual drive system.

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

The drive number may be omitted from the command. The current drive will be assumed. If you have a dual drive system you can format a disc in the second drive while the utilities disc remains in the first drive.

\* TWINCAT

This command doubles the number of files available from 31 to 62. This alternate catalogue can be accessed using the command \*SC

## **\*VERIFY (<DRV>)**

### **PURPOSE**

Checks each sector of a disc for legibility. It is done automatically when you use the \*FORM40 and \*FORM80 commands

### **EXAMPLE**

**\*VERIFY 1**

verifies drive 1.

### **DESCRIPTION**

Sector verification program

### **ASSOCIATED COMMANDS**

**\*FORM40**

**\*FORM80**

## 7 RANDOM ACCESS FILES

One of the major advantages of a disc over a cassette tape is that the read-write head of the disc can be moved to a specific place on the disc 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 Disc 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 Disc 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 the disc files are **EXT#** and **EOF#**. **EXT#** tells you how long a file is, **EOF#** returns a value of **TRUE** (-1) if the end of the file has been reached and **FALSE** (0) if not. All the BASIC keywords used to manipulate disc files are explained in the USER GUIDE. They are:

**OPENOUT**  
**OPENIN**  
**PTR#**  
**EXT#**  
**INPUT#**  
**PRINT#**  
**GET#**  
**PUT#**  
**EOF#**

To prepare a file to receive data the **OPENOUT** keyword is used. In the USER GUIDE the following example is given:

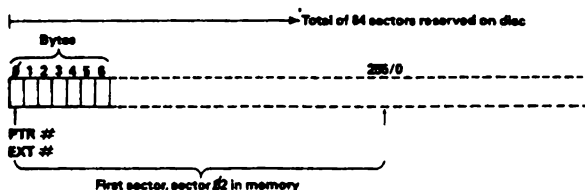
```
330 X = OPENOUT ("cinemas")
```

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 disc catalogue 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 disc 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 disc 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 10 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 "1". "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 in the table below.

"t" =&00 = String type, followed by "1", 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 "1" gives the length of the string in hex. The integer and the real number types are of fixed length as indicated above so they do not require the byte represented by "1" 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 the 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 disc and loaded sector 03 into the buffer to continue. This is still assuming that there are no other files on the disc, otherwise different sectors would be used. Remember that sectors 00 and 01 are reserved by the disc filing system for the disc catalogue. 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 disc. 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 disc if we want to. **OPENIN** is the BASIC keyword used to do this, e.g:

```
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 "cinema" 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 reads each cinema 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 disc 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:

```
10X=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 discs. 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 disc. If you want to create a random access file larger than 64 sectors on a disc with other files already on it, there is another method.

First save the file with the names you want 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 a disc 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 5 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 disc 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.

## 8 USING THE FILING SYSTEM IN ASSEMBLER

Section 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 summarised and particular use of **OSWORD** is described in detail.

### GENERAL PRINCIPLES

There are a number of routines available to handle disc 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 &FF00 to &FFFF. They then can call an internal (ROM resident) routine whose address is stored in RAM between &0200 and &02FF. 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 &021C. &021C and &021D contain the address of the executable routine in the disc file 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 disc files. The relevant routines together with their entry points are as follows:

<b>OSFIND</b>	<b>&amp;FFCE</b>	<b>FINDV</b>	<b>&amp;021C</b>	Open or close a file
<b>OSARGS</b>	<b>&amp;FFDA</b>	<b>ARGSV</b>	<b>&amp;0214</b>	Load or save data about a file
<b>OSFILE</b>	<b>&amp;FFDD</b>	<b>FILEV</b>	<b>&amp;0212</b>	Load or save a complete file
<b>OSBGET</b>	<b>&amp;FFD7</b>	<b>BGETV</b>	<b>&amp;0216</b>	Read a single byte to A from the file
<b>OSBPUT</b>	<b>&amp;FFD4</b>	<b>BPUTV</b>	<b>&amp;0218</b>	Write a single byte from A to the file
<b>OSGBPB</b>	<b>&amp;FFD1</b>	<b>GBPBV</b>	<b>&amp;021A</b>	Load or save a number of bytes
<b>OSWORD</b>	<b>&amp;FFF1</b>	<b>WORDV</b>	<b>&amp;020C</b>	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 indirections via &021C. The value in A determines the type of operation.

A=0 causes a file or files to be closed

A=&40 causes a file to be open for input (reading)

A=&80 causes a file to be opened for output (writing)

A=&C0 causes a file to be opened for input and output (random access)

If A=&80 or &C0 then Y (high byte) and X (low byte) must contain the address of a location in memory which contains the file name terminated with Carriage return (&0D). On exit Y will contain the channel number allocated to the file for all future operations. If Y=0 then the operating system was unable to open a file.

If A=0 then a file, or all files, will be closed depending on the value of Y. Y=0 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=0. The interrupt state is preserved, however 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 your 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=0      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=0 will return, in A, the type of file system in use. The value of A on exit has the following significance

- 0 no file system
- 1 1200 baud cassette file system
- 2 300 baud cassette file system
- 3 ROM pack file system
- 4 Disc file system
- 5 Econet file system
- 6 Teletex/Prestel Telesoftware file system

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

A=&FF "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=0. 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 by &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 the 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**

00	Address of file name, which must be terminated	LSB
01	by &0D	MSB
02	Load address of file	LSB
03		
04		
05		MSB
06	Execution address of file	LSB
07		
08		
09		MSB
0A	Start address of data or write operations,	LSB
0B	or length of file for read operations	
0C		
0D		MSB
0E	End address of data, that is byte after	LSB
0F	last byte to be written or file attributes.	
10		MSB
11		

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

A=0	Save a section of memory as a named file. The files catalogue information is also written.
A=1	Write the catalogue 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 files catalogue 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 8 bits have the following meanings:

Bit	Meaning
0	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 names are as follows:

0	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=0. The interrupt state is preserved but interrupts may be enabled during the operation.

#### **Read/write one byte**

OSBGET call address &OFFD7 to get a byte

OSBPUT call address &OFFD4 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 &OFFD1.

This routine will read or write 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 (lo-byte) and Y (hi-byte) point to the instruction block:

#### Offset

0	-----
	Channel
1	-----
	Pointer to data
5	-----
	Number of bytes to transfer
9	-----
D	-----
	Byte offset in file if used
	-----

A determines the type of operation:

A = \$01 Put bytes using byte offset

A = \$02 Put bytes ignoring byte offset

A = \$03 Get bytes using byte offset

A = \$04 Get bytes ignoring byte offset

This method is particularly useful in the environment of the **ECONET**<sup>§</sup> file system where the "packaging overhead" for transferring small amounts of data is proportionally high.

#### On Exit

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.

#### Read/write a sector

OSWORD with A=\$7F

Call address at \$0FFF1

A=\$F7 indicates that a general read/write operation is required.

On entry X (lo-byte) and Y (hi-byte) point to the instruction block:

#### Offset

0	-----
	Drive number
1-4	-----
	Start address in memory of source or destination of the data
5	-----
	Number of parameters
6	-----
	Command
7	-----
onwards	Parameters
	-----

<sup>§</sup> **ECONET** is a trademark of Acorn Computers Ltd.

### **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**

0 in the last parameter address +1 indicates a successful transfer. A failure is indicated by a disc error number.

## 9 CHANGING FILING SYSTEMS

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

*TAPE3	300 baud cassette
*TAPE12	1200 baud cassette
*TAPE	1200 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.



## 10 ERROR MESSAGES

### **&CC Bad filename**

This message appears if you enter a filename which is invalid - such as, longer than 7 characters, etc.

### **&D6 File not found**

The Disc filing system could not find the named file in the specified drive/directories.

### **&C3 File 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.

### **&C8 Disc changed**

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

### **&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.

### **&FE Bad command**

This means that the \* was omitted or that the command name was not recognised as a Disc filing system command or utility.

### **&CE Bad directory**

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

### **&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

### **&C6 Disc full**

This indicates that there is not enough space on the disc to open (OPEN-OUT?) or save a file of the specified size.

### **&BE Catalogue full**

The catalogue has enough space for 31 files. This error is produced if you attempt to enter more than 31.

### **&C4 File exists**

This occurs if you try to rename a file with an existing filename.

### **&C9 Disc read only**

This error occurs if you attempt to write to a disc which has the write-protection notch covered.

**&C1 file read only**

This error occurs if you try to write to a file opened for reading only, using OSFIND with A=40.

**&C7 Disc fault NN at TT SS**

If this error message occurs it means that the computer cannot read the disc. It implies that the disc is damaged, faulty, unformatted or of the wrong type, like an 80 track drive. NN is the error number TT is the track number SS is the sector number.

**&C5 Drive fault NN at TT SS**

This error means that the disc drive is probably faulty and needs attention.

**&C0 Too many open files**

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

**&CA Bad sum**

Bad checksum. For each random access file opened a control block is held in memory. This includes a checksum. A bad sum indicates corrupt data in memory and possibly a memory fault.

**&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.

**&ED Not enabled**

An attempt is made to use a restricted command without typing \*ENABLE immediately preceding it.

**&C2 File open**

This error occurs if you attempt to open a file which is already open. An intervening CLOSE is required between TWO OPEN 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.

## 11 TECHNICAL INFORMATION

## 18 BIT ADDRESSING

The BBC disc filing system uses 18 bit addressing giving a range from \$00000 to \$3FFFF. This means that two bytes and two bits are required to store a complete address, like this:

63FFFF 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 catalogue. This is clearly shown by the way that the disc catalogue is loaded into memory.

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

## DISC CATALOGUE

Sectors 00 and 01 on the discs are reserved for the catalogue. The format of the catalogue is as follows:

## Sector 00

**£00 to £07      First 8 bytes of the 13 byte disc title**

608 to 60E	First filename
60F	Directory of first filename

&10 to &1E	Second filename
&1F	Directory of second filename

..

**Repeated up to 31 files**

## Sector 01

£00 to £04	Last 5 bytes of the disc title
£05	The number of catalogue entries multiplied by 8.

```
#06 (bits 0,1)  Number of sectors on disc (2 high order bits of 10 bit
                number)
                (bits 4,5)  !BOOT start-up option
```

```

#07      Number of sectors on disc (8 low order bits of
          10 bit number)

```

508	First file's load address, middle order bits
509	First file's load address, low order bits
50A	First file's exec address, middle order bits
50B	First file's exec address, low order bits
50C	First file's length in bytes, middle order bits
50D	First file's length in bytes, low order bits
50E (bits 0,1)	First file's start sector, 2 high order bits of 10 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
50F	First file's start sector, 8 low order bits of 10 bit number
	:
	:
	:

repeated for up to 31 files.

Note that the first 8 bytes in each sector contain miscellaneous information about the disc. Each subsequent block of 8 bytes contain information about the files, repeated for up to 31 files. The complete information about file 3 would be found in the fourth block of 8 bytes on sector 00 followed by the fourth block of 8 bytes on sector 01.

#### FILE SYSTEM INITIALISE AND !BOOT

On pressing **BREAK** the MOS (machine operating system) seeks and initialises a filing system. It starts with the service ROM closest to the edge of the main printed circuit board. The first one will be initialised if just **BREAK** is pressed. If another key is held down while you press **BREAK** the first file system which recognises the key will be initialised. If none recognise the key, the CFS (cassette filing system), is initialised.

Having initialised a filing system, if "link 5" (see below) is unmade 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 disc 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 5

There are 8 links at the right-hand front of the keyboard, inside the machine. Numbering 1 to 8 from left to right, 5 determines the action of the selected file system. The options are described above.

## **THE FLOPPY DISC DRIVE PARAMETERS**

Several different disc drives may be used with the BBC computer. The same links on the front of the computer's keyboard (as mentioned above) are used to do this. Links three and four must be correctly set for each type.

Tandon disc drives with 4 msec. access require both links made.

Tandon and Shugart disc drives with 6 msec. access require 3 made 4 unmade.

MPI disc drives require link 4 made and 4 unmade.

Olivetti disc drives require both links unmade.

## **IMPORTANT**

The Tandon disc drives momentarily "whirr" when you close the drive door, so as to centre the disc on the rotating cogs. Unfortunately they rotate faster than normal, so that if writing is commenced the catalogue is corrupted. Do not initiate a writing action until the drive has settled.

## 12 FILING SYSTEM COMMAND SUMMARY

**COMMAND** (minimum abbreviation, full-stop required)

### **PURPOSE**

<b>*ACCESS (*A)</b>	Locks or unlocks a file
<b>*BACKUP (*BAC.)</b>	Copies all information from one disc to another.
<b>*BUILD (*BU.)</b>	Causes all subsequent keyboard entries to be stored in the named file.
<b>*CAT (*.)</b>	Displays the disc catalogue.
<b>*COMPACT (*COM.)</b>	Collects all files on a disc together into one contiguous sequence leaving a single block of free space.
<b>*COPY (*COP.)</b>	Copies all specified files from one disc to another.
<b>*DELETE</b>	Removes a single named file from the disc.
<b>*DESTROY (*DES.)</b>	Removes specified files from a disc in a single action.
<b>*DIR (*DI.)</b>	Changes the current set directory.
<b>*DRIVE (*DR.)</b>	Changes the current set drive.
<b>*DUMP (*DU.)</b>	Produces a hexadecimal listing of a file.
<b>*ENABLE (*EN.)</b>	Allows the use of <b>*BACKUP</b> and <b>*DESTROY</b>
<b>*EXEC (*E.)</b>	Reads a disc file byte by byte as if the bytes were being typed on the keyboard.
<b>*HELP (*H.)</b>	Displays the file system commands with syntax guidelines.
<b>*INFO (*I.)</b>	Displays information about specified files.
<b>*LIB (*LIB)</b>	Selects the drive/directory for the library.
<b>*LIST (*LIST)</b>	Displays a text file on the screen with line numbers.
<b>*LOAD (*L.)</b>	Reads a file from disc to memory.
<b>*OPT 1 (*0.1)</b>	Switches screen messages which accompany disc accesses on or off.
<b>*OPT 4 (*0.4)</b>	Specifies the auto-start option of a disc relating to a file named :0.\$.!BOOT.
<b>*RENAME (*RE.)</b>	Changes a filename.
<b>*RUN (*R.)</b>	Runs a machine code program.

<b>*SAVE (*S.)</b>	Saves a specific part of memory to the disc.
<b>*SPOOL (*SP.)</b>	Transfers all text subsequently displayed on the screen into a specified file.
<b>*TITLE (*TI.)</b>	Changes the title of a disc.
<b>*TYPE (*TY.)</b>	Displays a text file on screen with line numbers.
<b>*VERIFY (*V.)</b>	Checks legibility of every sector on a disc.
<b>*WIPE (*W.)</b>	Removes specified files from the disc after confirmation for each file meeting the given specification.

# INDEX

Abbreviations	69	Formatting	14,50,51
*ACCESS	21	*FORM40	50
Addressing	66	*FORM80	51
afsp	21		
Assembler	57	Getting going	7
Auto-start	18	Grandfather, father, son	11
*BACKUP	22	*HELP	35
BGET#	53		
BPUT#	53	*INFO	36
*BUILD	23	INPUT#	53
Bytes	14	Integer type	54
*CAT	24	Library	19
Catalogue format	66	*LIB	37
Changing filing system	63	*LIST	39
Commands	20	*LOAD	40
Command abbreviations	69		
Command summary	69	Multi-file operations	17
*COMPACT	25		
*COPY	27	OPENIN	53
Copying- diagram	13	OPENOUT	53
- master disc	11	*OPT1	41
- single drive	12	*OPT4	42
*DELETE	28	PRINT#	53
Demonstration programs	7	PTR#	53
*DESTROY	29	Random access	53
*DIR	30	*RENAME	43
Directories	17	Real type	54
Disc drive	3	*RUN	44
Disc drive parameters	68		
Disc files	16	*SAVE	45
Disc filing system	4	Sectors	14,15
Discs	9	Single drive copying	12
*DRIVE	31	*SPOOL	46
Drive numbers	16	String type	54
*DUMP	32		
*ENABLE	33	*TITLE	67
EOF#	53	Tracks	14,15
Erasure Prevention	11	*TYPE	48
Error messages	64		
*EXEC	34	Utilities	50
EXT#	53	*VERIFY	52
Filename	17	"Wildcard"	17
Files	16	*WIPE	49
File specifications	16	Write-protection	11
File system initialising	67		
File types	54		
Fitting a disc system	7,8		