Chapter 8

USE OF THE DISC FILING SYSTEM ON THE BBC MICROCOMPUTER

8.1 INTRODUCTION

The BBC microcomputer is capable of supporting a number of different filing systems for the interchange and storage of data and programs. All BBC microcomputers can support the cassette tape filing system. Additional firmware (ROMs) and hardware is necessary to support the floppy disc filing system, the Econet filing system, and the other filing systems available. The filing systems take care of the ‘house-keeping’ necessary to locate a file and allow the computer to transfer the contents of a file from the disc to the microcomputer. This takes only a few seconds, unlike the cassette filing system which may search a whole 30 minute tape until it either finds the file or runs out of tape.

In this chapter we are concerned with disc filing and it is assumed that a BBC microcomputer with an Acorn disc interface and at least one floppy disc drive is available. A dual floppy disc drive will be found convenient, and, in some applications, highly desirable. Some of the commands discussed here are common to all the filing systems whilst others are only available in disc systems. There are several other suppliers of disc interfaces and disc drives for the BBC microcomputer and in general these have common features with that of the Acorn system discussed here.

8.1.1 Discs and disc drives

In this type of storage system the recording medium is the magnetic coating of a rotating disc. Information is recorded along concentric tracks on the surface of the disc by a magnetic read-and-write head which can be positioned over any particular track. The high rotation speed and short head location time give access times which are very short compared with those provided by magnetic tape cassette. The favourable environment provided by the disc surface gives improved recording reliability. Each track is one bit wide and the data is recorded serially along the track.

The type of recording disc most frequently encountered on BBC microwmputer systems is a type known as a floppy disc, the name arising from the flexible nature of the mylar backing material. The mechanism which drives and accesses the disc is known as a disc drive and the recording disc is sometimes referred to as a diskette.

The diameter of the disc used in systems controlled by the BBC disc filing system software is 5¼ inches. The disc is housed within a permanent stiff plastic casing (see Figure 8.1) which incorporates holes for readlwrite head access, an index mark and a write-enable slot which may be covered as a protection against any further writing to the disc. Discs must be inserted into the drive with the label on top and with the head access slot at the back of the drive. The disc is driven at the centre by a cone and plate which engages and aligns the disc when the flap or lever on the disc drive is placed in the closed position. When not in a disc drive, discs should be returned to their protective covering envelopes, as a small speck of dust or a greasy fingermark on the surface of the disc may cause an error (‘Disc fault’) when that track is read. This is similar to the good practice recommended for tape and video cassettes.

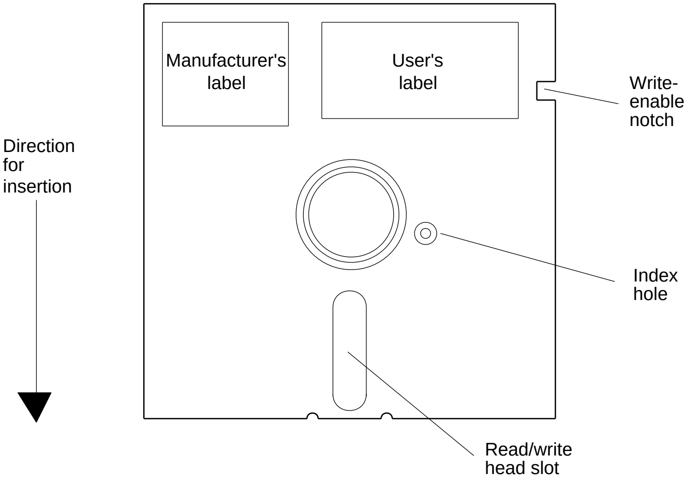

Figure 8.1 Features of a floppy disc.

The BBC disc system software permits the use of either 40 or 80 track disc drive units. The tracks are concentric circles spaced either 48 or 96 to the inch. The recording and reading head can moved to the appropriate track and has a total movement of slightly less than one inch to permit 40 or 80 tracks on a disc. Each track is divided into 10 sectors (see Figure 8.2). The Intel 8271 disc interface controller used in the BBC microcomputer only permits single density (as opposed to double density) recordings, and at the rotation speed of 300 RPM this determines a storage capacity of 256 bytes per sector. The storage capacity of a single-sided 40 track disc is therefore 10×256×40 bytes.

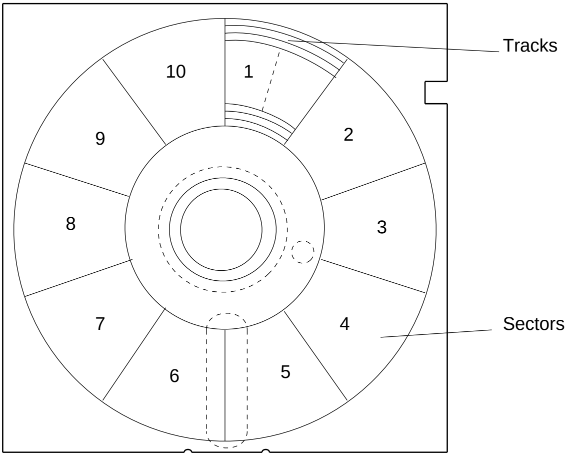

Figure 8.2 Organization of tracks and sectors on a floppy disc.

The term used for 1024 bytes is the kilobyte, so the corresponding storage capacity is 100K. A double-sided 80 track disc would have a storage capacity of 400K. Some disc drive units are switchable between 40 and 80 track operation. This is convenient when transferring files from one type of disc irive to another.

8.1.2 The disc filing system

Following power up a Model B microcomputer fitted with a Disc Filing System (DFS) ROM will normally display the message

```
BBC Computer 32K
ACORN DFS
BASIC
>
```

indicating the selection of the DFS as the default filing system. This message will also appear after <CTRL-BREAK>, and after <BREAK> the message is the same apart from the omission of the memory data (32K). Default conditions are those obtained when no specific action is taken to obtain other conditions. In the case of BBC microcomputers set up so that other filing systems (Econet for example) have default priority, the default conditions can be overridden to select the the DFS by pressing <CTRL-D> and <BREAK> together. Likewise <CTRL-T>+<BREAK> will select the tape filing system. This can be verified on any machine, after a <CTRL-T>+<BREAK>, by typing the *CAT command and observing that the cassette motor light on the keyboard comes on. Pressing <BREAK> will normally return the machine to the default of the DFS.

Once a BBC microcomputer is switched on, any of the available filing systems fitted can be entered by typing the appropriate operating system command, which in the case of the DFS is *DISC or (for transatlantic cousins) *DISK, and can be abbreviated to *D. for convenience.

Two main versions of the DFS have been released by Acorn. The first was DFS 0.90 on an 8K EPROM and an enhanced version 1.20 is supplied on a 16K EPROM when a second processor is fitted. This later 16K EPROM also contains the Econet network filing system software (NFS) which would otherwise occupy a separate ROM socket. The most apparent difference is the more rapid file transfer under DFS 1.20. Data files are handled by DFS 1.20 in about half the time taken by DFS 0.90.

8.1.3 Getting started

Once the DFS has been selected, the BASIC (interpreter) commands SAVE and LOAD will be available for files on disc in the same way as they were for other filing systems. To find the names of files stored on a disc the command *CAT may be used (for a catalogue of the disc *CAT has the abbreviation *.). Any BASIC program may be loaded by

```
LOAD "<file>"
```

where <file> is a valid filename.

Any BASIC program stored in the computer can be copied to disc by the command

```
SAVE "<file>"
```

where <file> is the filename. (This command will overwrite any previous version of <file>.)

8.2 STRUCTURE OF INFORMATION ON A DISC

8.2.1 Files, directories and libraries

Information is stored on disc as a series of files, each identified by a unique filename. A file can be any program, data or other collection of information stored on the disc. Filenames are limited to not more than 7 characters or digits or both. Reserved characters such as * # . or : must not be included in the filename since these are used in special commands. The Acorn DFS does not distinguish between upper and lower case letters in filenames. All filenames and operating system commands can be in upper or lower case. The case used for a file name by the computer is that used when it is created, and either case may be used subsequently in commands given by the user.

Files are stored in sets known as directories, which are identified by any single character except * # . : or a space. On power up or following a <CTRL-BREAK> the DFS automatically selects the $ directory (it defaults to the directory $): $ is said to be the *set directory*. It is not necessary to create a directory − they all exist but are empty until files are put in them. Directories on the DFS are really no more than prefix characters to file names, with $ having the special role of default directory and default library directory. The set directory can be changed to directory A (for instance) by use of the directory command

```
*DIR A
```

Following this, any commands to SAVE or LOAD programs will be assumed to refer to directory A (unless otherwise specified in the filename). For example

```
SAVE"PROG1"
```

will save the program as A.PROG1 and

```
LOAD" PROG2"
```

will load the program A.PROG2.

You might also have a program called PROGI on the same disc in directory B. As this would be saved as B.PROG1 there is no risk of confusion. In general you may find it useful to store sets of programs or other files under different directories so that their common identity is recorded. This is aided by the fact that files belonging to the same directory are listed together in a *CAT listing.

In addition to the set directory, the DFS also allows for a set library. This is for use with the *RUN <file> or *<file> commands which will cause the specified program to be selected first from the current set directory if it exists, or if not, the currently set library will be searched for the specified file. The set library is one of the directories on the disc, the default setting being the $ directory. It is conventional to use the $ directory as the library directory as it is set by default. However, any other directory, for instance & which is conventionally chosen as the library directory on level 1 Econet, can be set as the library directory by

```
*LIB &
```

The advantage of having a selected library is that the *RUN command can be used to run programs from this library whilst some other directory is selected for all other purposes.

8.2.2 Dual and double-sided disc drives

A double-sided disc drive counts as two disc drives as far as the DFS is concerned. The selection of a drive, if more than one is fitted, is done by the command

```
*DRIVE n
```

where n is a number with the value 0, 1, 2 or 3 (n cannot be a variable, it must be a number). For example selects

```
*DRIVE 1
```

selects drive 1. Drives 2 and 3 correspond to the second sides of the discs in double-sided drives. The second side of the disc in drive 0 is called drive 2 and the second side of that in drive 1 is called drive 3. If a single double-sided drive is fitted the first side is drive 0, the second side is drive 2 and there is no drive 1.

The *CAT command has an extension which enables you to catalogue any drive, whether selected or not. This has the form *CAT n where n is one of the integers 0, 1, 2 or 3.

8.2.3 The catalogue of a disc

The current files on a disc can be obtained by the catalogue command

```
*CAT
```

(or *. in abbreviated form).

This will give a listing of the files to the screen (or currently selected output device) in the format shown in Figure 8.3.

```
*CAT
DISC-NAME  (34)
Drive 0            Option D (off)
Dir. :0.$          Lib.:0.$

    AFILE          BFILE
    CFILE          ZFILE

  N.NEWFILE       Z.ANOTHER
  Z.ZFILE
```

Figure 8.3 *CAT display.

The *CAT command yields the following information: in the first line, the user-specified title for the disc − which may be blank − and in brackets the number of times the disc has been accessed for writing (modulo 100).

In the second line, the currently selected drive number is given (0, 1, 2 or 3) and the Option setting for the disc.

In the third line, the currently set directory and library are given in the form :0.$. The directory is $ and the 0 refers to the drive selected. The : and .

are *separators* that indicate that drive and directory parameters follow. The full specification for a file includes this information.

The next line of the display is blank and then there follows the current set directory filenames in two columns in alphanumeric order (by ASCII code values). At the end of the current set directory the files in other directories are arranged in a similar manner by alphanumeric order of directories, the files appearing in the form

```
N.NEWFILE      Z-ANOTHER
```

indicating the directory to which they belong.

8.2.4 Complete file specification.

In systems including more than one drive the file specification may include drive number information. For example the command

```
SAVE":1.K.PROGRAM"
```

would save the file PROGRAM in the K-directory of drive I even if the currently selected drive is number 0. This means that to load the program file AFILE from the disc catalogue shown above the following commands are all valid when the current set directory is $ and the drive selected is 0.

```
LOAD":0.$.AFILE"
LOAD"$.AFILE"
LOAD":0.AFILE"
LOAD"AFILE"
```

If another directory is the set directory then only the first two will load AFILE from the $ directory. The last two will load AFILE from the current set directory if it exists, or fail with the message ‘Not found’. In multiple disc drive systems it is thus possible to load from or save to any directory on any disc drive by appropriate choice of parameters in front of the file name.

If the file exists in directory $ on disc drive 1 then it may be loaded by

```
LOAD":1.AFILE"
```

so long as $ is the current set directory, irrespective of the selected disc drive.

8.3 FORMATTING AND VERIFYING

The number of tracks on one surface of a disc is determined by the size and minimum movement of the read/write head which is driven by a stepper motor through a screw thread. The available aperture on the disc envelope permits a movement of the head of approximately one inch. There are two common minimum step sizes: 1/48 of an inch and 1/96, of an inch, determined by the choice of stepper motor/screw thread combination, the precision of manufacture and the head which is much smaller for the smaller step size and must therefore be more accurately located on a track. The two step sizes are usually specified as 48 tracks per inch (TPI) or 96 TPI and floppy discs are often certified for these track densities. The disc drives which step 1/48 of an inch can accommodate 40 tracks and should be used with discs certified for 48 TPI. Disc drives with a step size of 1/96 of an inch can accommodate 80 tracks and need discs certified for 96 TPI. Discs certified for 96 TPI can be used for 40 tracks. Some disc drives can be switched from 80 to 40 track use − this is done by doubling the minimum track step size − but note that they use a small 80 track head and are not really suitable for writing to 40 track discs.

The rate at which data is read or written is determined by the floppy disc controller (an Intel 8271 disc controller chip in the case of the BBC microcomputer). It is necessary for the floppy disc controller to remain in synchronization with the data being transferred. In the case of the Intel 8271 this is done by interposing a clock pulse between each data pulse. These clock pulses are in addition to the synchronization provided by the single index hole punched in the disc. Discs which are synchronized in this way using a single hole, and in which the size of the sectors is determined by the formatting routine, are said to be soft-sectored. The hard-sectored 8" diameter discs have a second ring of holes for the start of each sector. The majority of 5¼" discs are soft-sectored, and only soft-sectored discs would normally be used with the BBC microcomputer.

The user establishes the sector structure on a soft-sectored disc by formatting the disc. This process is carried out by a program which uses special commands to the 8271, the BBC’s floppy disc controller. During the formatting procedure, the appropriate synchronizing, address, data and checking bytes are placed on each track. The tracks are laid out in turn starting with the outermost which is track 0 and proceeding inwards. The tracks begin with a synchronization signal to identify the start of the track (in addition to the index hole). On each track, each sector begins with a synchronization signal to identify the beginning of a sector, and four bytes to identify the track number, side, sector number and sector size. The data bytes follow after further synchronization bytes. At the end of the data block a cyclic redundancy check, generated from the data, is stored for checking during loading.

The BBC microcomputer Model B+ has a formatter in the operating system which is absent from the Model B. However, all the suppliers of disc drives provide a formatting program. These differ in detail, but all carry out essentially the same task. We shall consider only the official Acorn/BBC formatter, which is used in essentialy the same manner whether from the utilities disc or from the operating system.

Assuming the use of a single-sided disc drive, the formatting process commences with the insertion of the utilities disc containing the formatter program (in the case of the Model B) and the command

```
*FORM40 0
```

is entered, which evokes the response

```
Disk formatter 1.05

Do you really want to format drive 0 ?
```

If the drive specification is omitted the response is

```
Disk formatter 1.05

Format which drive?
```

to which the answer should be 0, which will then evoke the response

```
Do you reatty want to format drive 0 ?
```

as before.

Before answering this question the utilities disc is removed and replaced by the blank disc (on the Model B). The response Y then initiates the formatting process, during which, as each track is formatted it is also verified and the track numbers of completed tracks are displayed. The success message

```
Disk formatted
```

will appear when the process is successfully completed. If it fails the message

```
Verify error
```

or

```
Format error
```

will appear. At the termination of the process the option of formatting another blank disc is offered. The answer N then terminates the formatting program.

The verification of a disc or track checks the disc for ‘readability’ by reading each sector and checking that its checksum is correct. Verification forms an integral part of *FORM40 and *FORM80, and is also available as a separate utility program on the utilities disc as the file VERIFY and again is a built-in command on the Model B+. If this disc is placed in the selected library drive then

```
*VERIFY <drive>
```

will cause verification of the specified drive. This checks that all the files on the disc can be read, and can be used for reassurance about the condition of a disc.

8.4 OTHER DFS COMMANDS

In this section, round brackets are used to indicate an optional part of the command, and angle brackets, < >, are used for a general term such as the number of a disc drive. For example *CAT (<drive>) means that the following are valid

```
*CAT
*CAT 0
*CAT 1
*CAT 2
*CAT 3
```

In the case of *CAT the catalogue of files on the current set drive will be displayed, whilst in the other cases the catalogue of the disc in the specified drive will be displayed.

Wildcards are used for multiple file commands. A wildcard is a symbol which may represent any character (in the same way that the joker does in a pack of cards). The ambiguous file specification indicates that precise specification of files is not necessary. A group of files can be represented by the use of the * or # symbols, where # represents a single unspecified letter and * represents any number of unspecified letters or none at all. Thus ## can be used to represent any two letters. Partial specification can be used to select a group of files. For example

```
A.*
```

will select all files in directory A.

```
MY*
```

will select all files beginning with MY such as MYPROG,MY ,MYKE whereas MY## will only select files of four letters beginning with MY.

```
*2*
```

will select all files containing 2 anywhere in the filename from first to last character.

The directory identifying letter can also be represented by a wildcard (either * or #) but the ‘.’ cannot be so represented. The * and # only act as wildcards for valid filename and directory symbols. Thus to search all directories for files containing a 2 anywhere in the filename the wildcard sequence is *. *2* or #.*2*.

The full range of file handling commands of the DFS and of the Operating System are given in Table 8.1. There are in addition certain file handling commands associated with BASIC which are treated separately in Chapter 9. It is most convenient to consider the commands in groups.

Table 8.1 Filing System (DFS) commands.

Command	*Abrv.*	*Parameters*	*Comment*
`*ACCESS`	`*A.`	`<afsp> (L)`	
`*BACKUP`	`*BAC.`	`<sredrv> <destdrv>`	Used for backup
`*BUILD`	`*BU.`	`<fsp>`	
`*CAT`	`*.`	`(<drv>)`	
`*COMPACT`	`*COM.`	`(<drv>)`	
`*COPY`	`*COP.`	`<sredrv> <destdrv> <afsp>`	
`*DELETE`	`*DE.`	`<fsp>`	
`*DESTROY`	`*DES.`	`<afsp>`	
`*DIR`	`*DI.`	`(<dir>)`	
`*DRIVE`	`*DR.`	`<drv>`	
`*DUMP`	`*DU.`	`<fsp>`	
`*ENABLE`	`*EN.`		Used with OS 0.90
`*EXEC`	`*E.`	`<fsp>`	
`*HELP`	`*H.`	`(DFS) or (UTILS)`	
`*INFO`	`*I.`	`<afsp>`	
`*LIB`		`:(<drv>).<dir>`	
`*LIST`		`<fsp>`	
`*LOAD`	`*L.`	`<fsp>`	
`*OPT`	`*O.`	`M(,N)orM( N)`	
`*RENAME`	`*RE.`	`<oldfsp><newfsp>`	
`*RUN`	`*R.`	`<fsp> (parameters)`	
`*SAVE`	`*S.`	`<fsp> <start> <end> <exec> <Load>`	
`*SPOOL`	`*SP.`	`<fsp>`	Without <fsp> to end
`*TITLE`	`*TI.`	`<disc-name>`	Up to 12 characters
`*TYPE`	`*TY.`	`<fsp>`	
`*WIPE`	`*W.`	`<afsp>`	

fsp = file specification
afsp = ambiguous file specification
drv = drive number
dir = directory character

Round brackets are used to indicate optional parameters

3.4.1 Backup and file transfer

It is essential to be able to make copies of discs containing important programs and data. This can be done with the *BACKUP command, which vas to be immediately preceded by *ENABLE (on DFS 0.90). This feature is provided to prevent the inadvertent overwriting of a disc which might otherwise occur during the backup sequence. To make an exact copy of a disc using a single disc drive (drive 0) the following sequence of commands is typed

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

The copying process is done in sections and requires frequent interchange of the source disc and the destination disc. It is advisable to cover the write enable notch on the source disc with a write-protect sticker so that the original disc cannot be overwritten accidentally. The new (destination) disc must have been formatted previously, and will be completely overwritten.

The process is much more straightforward when two disc drives are available, in which case the commands are

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

to copy the contents of the disc in drive 0, sector by sector, onto the disc in drive 1. It is usual to backup 40 track discs to 40 track discs, and 80 track to 80 track. It is, however, possible to backup 40 track discs to 80 track, but it is not possible to backup 80 track to 40 track discs.

The *ENABLE line can be omitted if DFS 1.20 is fitted since the response to

```
*BACKUP 0 0
```

is then

```
Copying from 0 to 0
Go (Y/N) ?
```

which is used to prevent unintentional backing-up. A Y response permits the backing-up to proceed, whilst N ends the effect of the command. The response to *BACKUP 0 0 on OS 0.90 is the error message

```
Not enabled
```

The *COPY command is available to transfer one or more individual files from one disc to another, and has the form

```
*COPY <source drive> <destination drive> <afsp>
```

For example to transfer a group of files containing the letters MY from a disc in drive 1 to a disc in drive 0 the form would be

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

The value of this command is that files can be copied onto discs that already have some programs on them without overwriting the existing programs and it can transfer between 80 and 40 track discs. The *COPY command has much to recommend it as it will not overwrite existing files, but since it transfers one file at a time it is slower than *BACKUP when a large number of separate files are to be moved from one disc to another. The *BACKUP command transfers all the information from one disc to another, ignoring the file structure, in as few stages as possible by using all the available memory on the computer. Both *BACKUP and *COPY will overwrite existing programs in computer memory.

8.4.2 File removal

When discs have been in use for some time, files will accumulate that are no longer required. A single, named, file can be removed by the command

```
*DELETE <fsp>
```

A group of files can be removed by the commands

```
*ENABLE
*DESTROY <afsp>
```

Thus

```
*ENABLE
*DESTROY *.BWJ*
```

would give a list of all the files starting with BWJ and give the prompt

```
Go (Y/N) ?
```

to be answered Y to confirm that all the files are to be deleted. Again *ENABLE is not necessary with DFS 1.20 which gives an extra prompt after the *DESTROY command thus

```
*DESTROY
Go (Y/N) ?
```

Another command to allow selective deletion of a group of files is

```
*WIPE <afsp>
```

The name of each file corresponding to the ambiguous file specification is displayed, and the user replies Y if it is to be deleted, N if not. For example

```
*WIPE BWJ#
BWJ1    :Y
BWJ2    :Y
BWJ3    :N
BWJ4    :Y
```

will delete files BWJ1, BWJ2, BWJ4 but not BWJ3. The command *WIPE does not need *ENABLE because each file is separately presented for approval before deletion.

8.4.3 Disc organisation: *INFO, *OPT 1 and *COMPACT

The catalogue command *CAT which gives a list of the names of the files and their access status has been described above. Further information about the files on the disc can be obtained by use of the command

```
*INFO <afsp>
```

The information displayed for each file is: directory name, status, execution address, load address, length, sector address.

For example

```
*INFO A.*

A.MENU L 001900 00801F 00003D 003
```

The numbers are in hexadecimal. This line shows that the file MENU is in directory A. The status L indicates that the file is locked to prevent accidental overwriting from occurring. The execution address is that normally found for a microcomputer fitted with DFS and the load address is the normal load address for a BASIC program.

This information can be made to appear each time there is any file access by use of *OPT command in the form

```
*OPT 1 1
```

The additional information can be suppressed by

```
*OPT 1 0
```

Although the DFS takes care of the storage location of each file and the catalogue of files on the disc, it does not always make the most efficient use of a disc. After files have been deleted, or when files are extended and then saved again under the same name, there will be gaps between one file and the next. A storage problem can occur when a file has others after it on the disc or when it is the last file and fills the disc. If this file is modified and if this makes it longer it may no longer be possible to save it back into the same space. This will cause the error message

```
Can't extend
```

or if there is no room on the disc

```
Disc full
```

These messages may be generated even when there are gaps between the files due to deleted files. In order to release this space for efficient use a special command must be used which shifts files up on the disc so that no vacant sectors are left between one file and the next. This command is

```
*COMPACT (<drive>)
```

When this command is used, the program currently in memory may be lost, as the memory is used in the movement of files. It will therefore be necessary to save any program in memory to another disc with free space before use of the *COMPACT command. Files always use an integral number of sectors, and any space not used by a file at the end of the last sector is left blank.

8.4.4 File protection

The advantages of backing-up discs and copying files are that secure copies of files can be made. Limited security is available on the disc currently in use by means of the *ACCESS command which permits files to be ‘locked’ against being overwritten or deleted. For example

```
*ACCESS A.FRED L
```

will result in the file FRED being locked. Files that are locked are followed by an L in the catalogue listing obtained using *CAT. To unlock the file to amend it, *ACCESS without the L may be used. For example

```
*ACCESS A.FRED
```

The *ACCESS command can be issued for a group of files with the ambiguous file specification. For example

```
*ACCESS A.* L
```

locks all files in the A directory.

Once locked, a file is not affected by the DFS commands *SAVE, *DELETE, *DESTROY, *WIPE, *RENAME or the BASIC command SAVE "<file>". The * ACCESS command offers no protection to the *BACKUP and *FORMAT commands. The only protection against *BACKUP and *FORMAT is to use a write-protect label which protects the complete disc. The use of the write-protect label on the backup disc is recommended where valuable programs or data are stored on a disc. It should be noted that these precautions cannot protect a file from being physically corrupted, for example by dust or by a magnetic field (which might occur when a disc is left in the disc drive when it is switched on or off) and for these reasons separate backup copies of all important files should be kept.

8.4.5 Other DFS organization commands

***RENAME**

The name of a file is usually chosen so that it reflects in some way the contents of the file. When new files are created it is sometimes found that there is a conflict between the name of an existing file and that chosen for the new file. In these circumstances it is convenient to be able to change the name of a file; this can be done with the command

```
*RENAME <old fsp> <new fsp>
```

This command changes name of the file <old fsp> to <new fsp>.

*TITLE

Each file on a disc has a name and they can be grouped into different directories. It can also be useful to give a name to the disc. This can be done by the command

```
*TITLE <disc name>
```

The command permits a title of up to 12 characters long to be appended to the disc in the currently selected drive. The title will then be displayed whenever that disc is catalogued.

*HELP

There are occasions when it is useful to be able to find out certain characteristics of the particular microcomputer that you are using. This can be done with the command

```
*HELP
```

The command displays information concerning the ROMs fitted within the BBC microcomputer.

Other *HELP options

On other occasions a quick revision of the available commands can be useful, in which case the following modifications to *HELP will be found helpful

```
*HELP DFS
```

which dislays a list of the DFS commands with their syntax. Similarly

```
*HELP UTILS
```

displays the syntax of the DFS commands concerned mainly with the creation and listing of ASCII files.

If other ROMs are fitted in your computer, they may provide further *HELP options. For example, all the commands of the popular utility ROM *Disc Doctor* can be listed by *HELP DISC DOCTOR.

8.4.6 Machine code programs

The commands *SAVE, *LOAD and *RUN are associated with machine code programs. They are analogous to SAVE, LOAD and CHAIN and are discussed more fully in Section 10.8.5.

8.4.7 ASCII file commands

These commands are

```
*SPOOL <fsp>
*EXEC <fsp>
*BUILD <fsp>
*TYPE <fsp>
*LIST <fsp>
```

and they are appropriate to files which consist only of a string of ASCII characters. This does not include the normal BASIC program files in which the BASIC keywords are replaced by single tokens.

```
*SPOOL <fsp>
```

enables the user to send all that appears on the screen to an ASCII file. This can be the listing of a BASIC program, for instance. The output to the file is terminated by the command

```
*SPOOL
```

without any file name, or by the use of

```
CLOSE#0
```

which closes all open files. Once a BASIC program has been put into an ASCII file it can be accessed by a word processing system for editing purposes, or it can be merged with other sections of text or with other BASIC programs. If the ASCII file is subsequently required for use as a program again then the *EXEC command can be used.

```
*EXEC <fsp>
```

will cause the contents of the ASCII file specified to be treated just as though it were direct keyboard entry.

The use of the *SPOOL and *EXEC commands is described in detail in Section 9.7.

```
*BUILD <fsp>
```

enables text to be entered into an ASCII file directly from the keyboard.

The text in a file can be displayed by the use of

```
*TYPE <fsp>
```

which will cause the ASCII text version of the file to be sent to the screen. This should only be used with ASCII files as the codes stored in BASIC and other numeric data files may corrupt parts of the computer memory and (temporarily) disable the computer.

Another format for the display of ASCII files is obtained with

```
*LIST <fsp>
```

which gives a numbered listing of the ASCII file. Each line is numbered sequentially as it is displayed.

The complete contents of any file, whether binary, program or text, can be displayed with

```
*DUMP <fsp>
```

which displays each byte of the file in hexadecimal and also gives the ASCII character where possible.

8.5 TURNKEY SYSTEMS

In some situations it is convenient to. start the execution of a program, often the MENU program for a complete disc, by a single keyboard operation. A program or a suite of programs is called a turnkey system if all the user has to do is switch on and load the disc for the software for his problem to be in place and ready to run.

The BBC microcomputer provides an auto-boot facility for which the required initialization action is to press <SHIFT-BREAK> and then release the <BREAK> key first. Once started, the auto-boot facility interacts with the file !BOOT, if it exists, on the disc in drive 0. The form of the interaction depends on the option setting of the disc which may be 0, 1, 2 or 3. The use of the word boot here arises from early computers where in order to start them it was necessary to load a special program which enabled the system to get itself going. This process is analogous to the idea of lifting yourself by your own bootstraps, hence this was called the ‘boot’ program.

The disc option is set by the *OPT command followed by two numbers separated by spaces. The first first number should be 4 to indicate a change of disc option and the second number should be the required option 0, 1, 2 or 3. Table 8.2 summarizes the options.

Table 8.2 Effects of *OPT 4.

`*OPT 4` *option*	*Action*
`0`	`Normal` <BREAK>
`1`	`*LOAD !BOOT`
`2`	`*RUN !BOOT`
`3`	`*EXEC !BOOT`

The action following <SHIFT-BREAK> depends upon the previous setting of the disc option. For example

```
*OPT 4 0
```

sets the disc option to 0. The 0 option means that the auto-boot facility is switched off and <SHIFT-BREAK> behaves like <BREAK>.

Disc option 1 will cause the file !BOOT to be *LOADed on operation of the auto-boot facility. If the disc option is set to 2 the !BOOT file will be *RUN. The file !BOOT will need to be a machine code program for this to be used. Probably the most widely used setting of the option is 3, which causes the !BOOT file to be treated as though the *EXEC command had been given for it.

The correct operation of the auto-boot action is conditional on the existence of the appropriate !BOOT file on the disc and on the setting of the appropriate *OPT 4 command. It is perhaps unfortunate that there is not an option to CHAIN"!BOOT" as a BASIC program, since this is very frequently what is really required.

For turnkey systems, the only way to auto-boot a menu or other start-up program is to use disc option 3. To use this option the !BOOT file needs to contain an ASCII version of the immediate mode commands to start the system. An ASCII !BOOT file will normally be created by the use of the *BUILD command.

For example, consider the operation of the command *BUILD !BOOT. As each line is typed, it will be given a line number, and on completion, pressing the ESCAPE key will transfer the file to the current drive. The !BOOT file would be created by the sequence of commands

```
*BUILD !BOOT
```

1. `*TV 0,1`
2. `CHAIN "MENU"`
3. <ESCAPE>

If the disc option is now set by the line

```
*OPT 4 3
```

then on operation of the auto-boot by pressing <SHIFT-BREAK> and releasing <BREAK> first, the *TV command to remove interlace of the video monitor is executed and the program "MENU" will be loaded and run.

In the creation of an EXEC file such as the above !BOOT file it is not necessary to follow the commands by |M (the symbols for <CTRL-M> which are equivalent to <RETURN>) as is done with the *KEY commands. The *EXEC command treats each line of the file as though it had been entered from the keyboard and followed by <RETURN>. Thus it would be possible to issue the command

```
*EXEC !BOOT
```

to start a menu driven system with the above !BOOT file and an appropriate MENU program.