HDFS

A Hierarchical Disc Filing System for the BBC Microcomputer

Angus Duggan

Contents

1 Introduction

The Hierarchical Disc Filing System, **HDFS**, which this manual describes is designed to work with the Intel 8271 Floppy Disc Controller chip fitted in older BBC microcomputers. It is compatible with the Acorn 1.20 DNFS,[1] and provides some extra functionality.

This manual will assume that the user is familiar with the operation of disc drives, and that the disc interface, disc drive(s) and ROM are installed (see appendix B for details of how to install the ROM). A simple introduction to floppy discs can be found in appendix A. Users who are familiar with the Acorn 1.20 DNFS should read the next section, which will outline the major differences between **HDFS** and the DNFS. Users not familiar with the DNFS should miss out this section on first reading.

2 Differences between DNFS and HDFS

The **HDFS** has some real advantages over the DNFS. These include:

Hierarchical directories The **HDFS** has a fully hierarchical directory structure, allowing more than the 31 files per disc side of the DNFS.

40/80 track switching If the **HDFS** is used with an 80 track disc drive, it can automatically detect 40 track discs, and read and write to these discs without manual intervention.

Advanced tube support The **HDFS** includes the Acorn Tube Support code, with the extra feature of being able to turn off the Tube from software.

Sideways RAM support All of the memory moving, loading and saving operations can access any sideways ROM socket, allowing multiple banks of sideways RAM to be supported.

Large files If the **HDFS** is used with double sided disc drives, it can split files across both sides of the same disc, allowing files up to 400k long.

Low memory loading The **HDFS** can load files into memory which would normally take too much space, automatically disabling itself after the file is loaded.

Open files The **HDFS** supports a total of six open files simultaneously, one more than the DNFS.

More commands The **HDFS** has many more commands than the DNFS, including a fully featured disc, file, and memory editor, disc and memory searching, memory block moving, and a lot more.

DNFS compatibility The **HDFS** is also able to read, write, and format DNFS compatible discs.

[1] The majority of programs designed to work under DNFS 1.20 will work under **HDFS** with no change. Where changes have been required, it has usually been because the program's author(s) have made invalid assumptions about command parsing or memory usage.

3 Disc organisation

Throughout this and the following sections, a distinction will be made whenever there are differences between the two supported file structures; the Acorn DNFS format, and the **HDFS** native format. In cases where no distinction is made, the file structures are the same.

3.1 Files

Data is stored in named *files* by the disc filing system. These files have certain features which are common across filing systems; they are accessible in the same way by a set of operating systems commands.

In both the DNFS and **HDFS**, files have certain features; they are held in contiguous areas on the disc, have the same attributes, and the maximum length of filenames is 7 characters.

A file's *attributes* are its length, start sector address, load address, execution address, and permissions.

The maximum length of a file is determined by the amount of free space available in the directory; this can be up to 200K for the DNFS, and up to 400K for **HDFS**. The DNFS allocates 18 bits to the file length, whereas **HDFS** allocates 19 bits.

The start sector address determines where the file is held, relative to the start of the directory. There are 10 bits allocated to the start sector by DNFS and 11 bits by **HDFS**.

The load address determines whereabouts in memory the file is placed when it is loaded. There are 18 bits allocated to the load address by both **HDFS** and DNFS; if the high-order bits (above 16) are all 1's, the low-order 16 bits are used to refer to an address in the I/O processor memory, if a second processor is fitted.

The execution address determines where the computer jumps to start a program that is run from the disc. As with the load address, 18 bits are allocated to the execution address, and if the high-order bits of the execution address are all 1's, the address referred to is in the I/O processor.

If there is no second processor fitted, the load and execution addresses always refer to the I/O processor.

A file's permissions determine whether the file can be read, written, executed, or deleted.

3.2 Directories

Directories are used for partitioning files.

In the DNFS, directory names are a single character prefix which may be used to sort the files into different types. The "current directory" is a single prefix character which is added to all filenames which do not specify a directory. There may be no more than 31 files per disc in the DNFS. The directory information for the DNFS takes two sectors (512 bytes).

HDFS supports hierarchical, nested directories. These directories are held in contiguous blocks, exactly like files. Directory names must be between 2 and 7 characters long, and appear like normal filenames in the directory listing. Each of these directories, as well as the disc's root directory, may contain 31 files and sub-directories. **HDFS** directories have all of the normal file attributes, and also a flag which marks them as directories rather than files.

The length of a directory determines how much space is available for files and subdirectories inside the directory. It will always be a whole number of sectors (a multiple of 256 bytes), and 2 sectors (512 bytes) will be used for the directory information.

The load and execution addresses are meaningless for directories.

The readable and writable permissions are meaningless for directories.

3.3 Permissions

The **HDFS** supports the full range of permissions for files; they may be marked as readable, writable, executable, and/or locked (not deletable). The characters `R`, `W`, `X`, and `L` are used to indicate these in directory listings.

The readable and writable permissions refer to the file-handling operations OSFIND, OSBPUT, OSBGET and OSGBPB. Unwritable files can still be saved to, and unreadable files can still be loaded.

HDFS directory permissions are the same as for files except that the readable and writable flags are meaningless. The executable flag has a special meaning; when a directory is executed, it is made the current directory, and the directory options are used to determine whether to search for and load, run, or `*EXEC` a file called `!BOOT`, in the same way as the auto-boot facility. This facility allows applications to be organised within directories, and set up and run by simply executing the directory.

The DNFS only supports the *locked* permission. In DNFS, locked files are unwritable, and all files are readable and executable.

3.4 Auto-booting

Both the DNFS and **HDFS** support the auto-boot facility; if SHIFT-BREAK is pressed, and BREAK is released first, the filing system will look at the root directory's options to determine whether to search for and load, run, or `*exec` a file named `!BOOT`. The values of the options for each of the possibilities is given in section 11.7.

To allow the **HDFS** and DNFS to co-exist in a machine, each filing system checks whether certain keys are held down before initialising on a BREAK. If there are no keys held down, the filing system in the higher priority socket will take control.

The DNFS will not initialise if any other key than D is pressed with BREAK.

HDFS recognises two keys, H and I. The H key causes a normal initialisation. The I key initialises the computer for I/O processor operation only, if a second processor is present. This has the same effect as turning the second processor off, in software. CTRL-BREAK must be pressed to re-activate the second processor.

4 Disc management

The following sections contain some tips about disc and file management.

4.1 Formatting and verifying

New discs need to be formatted before they can store data. This lays down the track and sector structure on the disc, so that data can be located and written or read.

The tracks on the disc are arranged in concentric bands, each containing 10 sectors of 256 bytes. The sectors are numbered from 0–9 within each track, but the sectors numbers along a radius on the disc are not the same; this is because the drive takes a fixed amount of time to step between tracks, during which the disc keeps turning. A few sectors are missed while the step occurs, so the disc would have to do another revolution to get back to the first sector. The discs are arranged with a *skew* to avoid this, so that the first sector will be next under the drive head after the drive has stepped. The skew value is normally 3, but disc drives which step from track to track very slowly may need higher skews to perform best, and fast disc drives may be able to use a smaller skew.

The built in formatter in **HDFS** allows discs to be formatted double or single sided, 40 or 80 tracks, and with an **HDFS** catalogue or an Acorn catalogue. Double sided and 80 tracks can be used on all but a few old disc drives. The drive characteristics can be set using the keyboard links described in appendix B. Note that **HDFS** assumes that 80 track drives are being used, and single stepping mode should be enabled if 40 track drives are being used (see 6.11).

The choice of Acorn or **HDFS** format catalogues depends on whether the data must be readable on a computer which is not running **HDFS**. Most disc filing systems support the Acorn catalogue format, and so it should be used for discs which will be used to interchange data between machines. The **HDFS** format should be used when a large number of files may be put on the disc, or when a larger contiguous free space is desirable, *i.e.*, for all other discs.

Never try to format a disc with more tracks or sides than it is rated for. Discs are all made by the same manufacturing process, and low density discs are simply discs which failed tests for high density storage. It is likely that discs formatted with more tracks or sides than they are rated for will lose their data.

After formatting a disc, it is a good idea to verify it to check that there are no errors. A small percentage of discs will not format correctly, and may need re-formatting. If a disc fails to format after a few tries, it should be discarded.

4.2 Loading and saving

All of the normal loading and saving commands used by language ROMs are supported by **HDFS**, as well as the operating system `*SAVE` and `*LOAD` commands. A file name must be given when using these operating system commands; no default name is assumed.

4.3 Copying and renaming

Files can be copied between discs and directories or duplicated using the `*COPY` command. The `*RENAME` command is almost identical, except that it deletes the original copy of the file. These commands use the I/O processor memory as a buffer while copying files, and so should not be used if you have valuable data in memory which has not been saved.

It is possible to copy files from disc to disc with one disc drive. In this case the computer will prompt for the source or destination disc each time it has finished reading or writing data. If the wrong disc is inserted, the transfer will be stopped with an error message. If this happens, or if the transfer is aborted, the disc catalogues will not be changed.

It is possible to copy or move files larger than the memory buffer with `*COPY` and `*RENAME`. They load as much as possible into the memory at a time, and so require the least number of disc changes possible.

There is no specific command for making backups of discs, but `*COPY` and `*RENAME` can be used to copy a set of files into a directory to achieve this end.

4.4 File handling

All of the normal file opening, closing, and get and put byte commands used by language ROMs are supported, as well as the operating system OSGBPB routine for transferring blocks of data to and from files. Full random access to files is possible, using the sequential pointer.

Files can be opened for input, output, or update (input and output). Files opened for input cannot be used for output, and files opened for output cannot be used for input. The **HDFS** only allows access for input or output if the appropriate read or write permissions are set. Acorn format discs always have read and write permission on files. The **HDFS** allows up to six files open simultaneously.

When a new file is opened for output, it is located in the largest gap in the directory to allow it space to grow. A maximum of &4000 bytes are allocated to that file immediately, and more space can be claimed if it is available later. If more than one new file is opened, the second file will probably be allocated immediately after the first file, limiting the first file's length to &4000. If several large files are to be created by a program, the `*CREATE` command or OSFILE entry &7 can be used to make sure that enough space is allocated for each of them.

When a file is opened for output with the same name as an existing file, an attempt is made to delete the existing file, and its space is re-used. Files which are opened for update or input must exist already.

If the sequential pointer of a file is moved past the end of the file, the file will be extended as necessary by padding with zero bytes.

The OSGBPB call should be used whenever possible, to reduce the overheads associated with accessing files.

4.5 Compacting and extending

After a while, discs tend to get fragmented, with lots of short gaps separated by files. The space can be recovered into one contiguous block using the `*COMPACT` command. This shuffles all of the files and directories down so that there are no gaps between them, leaving all of the space at the end.

Directories can be shrunk to fit the files that they contain using the `*COMPACT` command. This is especially useful when the files in a directory are not being changed at all, *e.g.*, for program directories which are finished, or games directories. The executable directory feature of **HDFS** can be used to make these into self-contained applications which can be `*RUN`.

If a file has run out of space to grow, the `*EXTEND` command can be used to move it into the largest gap on the disc, where it may have more space to grow. This command can be combined with `*COMPACT` for maximum effect; the sequence `*EXTEND`, `*COMPACT`, and then `*EXTEND` will ensure that the maximum amount of space left in the directory is available for the file to grow into.

Both the `*COMPACT` and `*EXTEND` commands overwrite the I/O processor memory, and so should not be used if you have valuable data in memory which has not been saved.

5 Command syntax

Some conventions are followed in the descriptions of the command syntax given. The syntactic elements in the descriptions are:

dsp This is a directory specification.

For DNFS directories, a directory specification is an optional drive specification followed by a single character directory name. If the drive specification is present, it should be separated from the directory name by the directory separator character (default '`.`').

An **HDFS** directory specification is an optional drive specification followed by a series of directory names, each separated by the directory separator character (default '`.`'). **HDFS** directory names are similar to filenames, but can be between 2 and 7 characters long. The drive specification refers to the root directory of the drive, and the directory names are relative to that point (or the current directory if no drive specification is given).

The root directory character (default '`~`') can be used to refer to the root directory of the current drive. The parent directory character (default '`^`') indicates the parent of the current directory. A sequence of parent directory characters separated by directory separator characters can be used to refer to a directory much higher in the hierarchy.

In both cases, if a drive specification is given, it must be immediately preceded by a drive prefix character (default '`:`').

The maximum length of a directory specification (including separator characters) is 31.

fsp This is a file specification. It consists of an optional directory specification, followed by a file name. The file name can be between 1 and 7 characters long, and should be separated from the directory specification by a directory separator character (default '`.`'), if one is present.

The maximum length of a file specification (including separator characters) is 31.

In some cases, a file specification can also refer to a directory; the descriptions of the commands below state when this is possible.

afsp This is an ambiguous file specification. An ambiguous file specification is the same as a file specification, but includes at least one wild card character. The wild card characters are the glob character[2] (default '`*`'), which matches any number of characters, and the query character (default '`?`'), which matches any one character.

[2] Named after its UNIX equivalent.

drv This is a drive specification. A drive specification is merely a number indicating which disc drive to use. Up to four surfaces are supported by **HDFS**, and the drive numbers for these are between 0 and 3. With double sided drives, drive number 2 and 3 refer to the reverse sides of drives 0 and 1. These drive numbers are not generally used with **HDFS**, which can treat both sides of a double-sided drive as one disc.

If a drive specification appears in an optional context, it must by prefixed by the drive prefix character (default ':').

load, exec, addr These are hexadecimal memory addresses. A nominal 32-bit addressing range is used; if the top 16 bits are &FFFF, the bottom 16 bits are used to address the I/O processor memory. If there is no second processor, all addresses always refer to I/O processor memory. I/O processor addresses in the range &8000–&BFFF refer to sideways ROM sockets; the ROM socket accessed by these addresses can be set with `*OPT 3`.

Only 18 bits of the load and execution addresses are stored in **HDFS** format directories, so many addresses above the stored range will produce the same results.

len This is a hexadecimal file length. **HDFS** stores 19 bits of the file length, whereas DNFS stores 18 bits.

start, (+)end, (+)offs These are hexadecimal addresses, usually of disc sectors, but sometimes memory addresses. If the optional plus signs are used for the end or offset sectors, they are treated as offsets from the start address.

Sector addresses in **HDFS** are 11 bits long, DNFS sector addresses are 10 bits long.

LXWR This is a permissions specification; any combination of the characters `L`, `X`, `W`, and `R` (including none of them) can appear.

string Strings are delimited by double quotes '`"`'. Any printing character can appear in a string, and the normal operating system escape sequences can be used to insert non-printing characters. The vertical bar '`|`' is used to make a control character from the character that follows it, the sequence '`|!`' sets the high bit of the character or control character that follows, and the sequence '`|?`' is used to represent the DELETE character (&7F).

The length of the string accepted is dependent on the context in which it appears.

hex Hexadecimal numbers are accepted by some commands, usually in a sequence of numbers. These can be 1, 2, 3, or 4 byte numbers.

dec Decimal numbers are accepted by some commands. These numbers should be positive, and less than 65536 (2 bytes).

skew This a hexadecimal number which is used to set the track skew when formatting discs. It should have a value between 0 and 9.

:~^.*? This specification is only used once, for the `*WILD` command. Six characters are required, none of which are the same as any of the others.

Any other numbers or single characters appearing in the syntax are literals, and should be specified exactly as shown.

Note that some file specifications may be valid for DNFS and **HDFS**, whereas others (which include directories) are valid only for DNFS or **HDFS**.

There are several "Meta-characters" which modify the syntactic elements:

{. . . } Curly braces indicate that the part of the description between the braces may be repeated as many times as necessary. There are limits on the maximum number of repetitions possible, which depend on the context in which it appears.

<. . . > Angle brackets are used to group alternatives (see the forward slash, below) which are not optional (*i.e.*, at least one of the alternatives must appear).

(. . .) The part of the description between the parentheses is optional, and may be omitted entirely.

. . . /. . . The parts separated by forward slashes are alternatives; one and only one of them should be specified.

6 Filing system commands

This section describes the filing system commands available. These commands are only active when **HDFS** is the current filing system. The syntax of each command is given in the section title.

All of these commands may be prefixed by "H" to distinguish them from similar commands in other ROMs.

The output of these commands cannot be *SPOOLed (the *SPOOL file is actually turned if output is generated by these commands).

6.1 *ACCESS afsp (LXWR)

This command changes the permissions on the files or directories specified.

6.2 *ATTRIB fsp (load exec) (len) (LXWR)

The attributes of a file can be changes with this command. The file's length cannot be made longer than the space available for it. As files are held in contiguous blocks, the space available is the number of sectors used by the file plus the number of sectors free immediately after it.

6.3 *BUILD fsp (A)

Simple text files can be created with this command. Lines are read in, and written to the file. The A flag indicates that the file should be appended to, rather than overwritten.

Due to space limitations, there is a limit of 63 characters per input line.

The execution address of files created with *BUILD is set to &FFFFFFFF, because it is assumed that they will be used for *EXECing.

6.4 *COMPACT (dsp) (D)

This command compacts the directory specified, or the current directory if no directory is given. Files are moved together so that the maximum amount of free space is left in the directory. If the D flag is given, the directory specified will be shortened so that there is no free space left.

This command overwrites the contents of the I/O processor memory, so it should not be issued if there is valuable data in memory which has not been saved.

6.5 *COPY <fsp fsp/afsp dsp> (C)

Files or directories can be copied between different directories or discs with this command. Files or directories can also be copied to a different name on the same disc with the command. There are two different forms for this command; if two filenames are specified, the contents of the first file will be copied into to the second. If an ambiguous file specification and a directory specification are given, all of the files matching the ambiguous file specification will be copied into the directory. Either a DNFS or an **HDFS** directory can be given, and files can be copied between **HDFS** and DNFS discs.

The C flag prompts to change the disc between reading and writing files, so files can be transferred to other discs with one disc drive.

The maximum amount of information is read or written each time the disc is changed, so the fewest number of disc changes possible will be used.

This command overwrites the contents of the I/O processor memory, so it should not be issued if there is valuable data in memory which has not been saved.

6.6 *CREATE fsp (load exec) (len) (LXWR) (D)

This command creates a new file entry or directory, but does not actually write any data into it. If the load and execution addresses and the length are not specified, they default to zero.

The D flag indicates that a directory is to be created; in this case, the length must be a whole number of sectors, greater than two. A new catalogue is initialised in the first two sectors of the directory. Note: when creating directories, at least one of the attributes LXWR should be given, otherwise the D flag may be mistaken for the directory's length. It is a good idea to make directories bigger than required initially, as they can be shrunk to fit the files easily later, and files can be shuffled around inside them.

The file or directory's length cannot be made longer than the space available for it.

6.7 *DEDIT (drv) start (+)end ((+)offs) (R)

This command invokes the disc, file, and memory editor on the section of the disc specified by the start sector, up to but not including the end sector. If an offset is supplied, the cursor will be placed at that point initially.

The R flag invokes the editor in read-only mode, which can be used to view the data but not change it.

If no disc drive is specified, the current drive is used.

6.8 `*DELETE fsp`

This command deletes the file or directory specified. Directories cannot be deleted unless they are empty. Locked files or directories cannot be deleted.

6.9 `*DFIND (drv) start (+)end <string/{hex}>`

Data on a disc can be searched for with this command. The sector and byte address of each occurrence of the data is displayed. The search performed by this command is optimal.

The data to search for is specified by a string or a sequence of hex numbers. The maximum length of the string or sequence of hex numbers is 31 bytes.

The area of the disc to search is delimited by the start and end sector addresses.

If no disc drive is specified, the current drive is used.

6.10 `*DIR (dsp)`

This command sets the current directory to the directory name specified. If no directory name is specified, it lists the contents of the current directory.

6.11 `*DRIVE drv (S/D/A) (C)`

This command sets the current drive, or affects the parameters of a drive.

If the `S`, `D`, or `A` flags are given, the drive is set into single step, double step, or auto-step mode. Single step mode should be used for 40 track drives; **HDFS** otherwise assumes that 80 track drives are used, and will try to double step. The double step mode allows 40 track discs to be read on 80 track disc drives. Normally auto mode is enabled, in which double stepping will be used for 40 track discs. The single or double-step modes may also be necessary when a disc has a copy protection scheme which alters the normal track layout or number of sectors per disc. The default value of the single and double step mode for all drives can be set with the keyboard links described in appendix B.

The `C` flag disables sector address checks, which normally prevent access to sectors outside the range on the disc. This may be needed to allow some elaborate copy-protection schemes to work.

6.12 `*DUMP fsp`

A hexadecimal dump of the file is displayed by this command, with an ASCII representation beside it. In 80-column display modes, 16 bytes are shown per row. In 40 and 20-column display mode, 8 bytes are shown. The dump shows the address of the row (relative to the start of the file), the byte values in hexadecimal, and then their ASCII representation. Non-printable characters appear as dots. Bytes past the end of the file appear as asterisks.

6.13 `*ENABLE`

This command does nothing; it is allowed for compatibility with DNFS.

6.14 *EXTEND fsp

This command ensures that the file or directory specified has the largest space possible to grow into, without shuffling other files around. This may involve moving the file itself. The sequence `*COMPACT`, then `*EXTEND`, and then `*COMPACT` again will usually create the maximum space possible for growth.

This command overwrites the contents of the I/O processor memory, so it should not be issued if there is valuable data in memory which has not been saved.

6.15 *FEDIT fsp (offs) (R)

This command invokes the disc, file, and memory editor on the file specified. If an offset is supplied, the cursor will initially be placed at that point relative to the start of the file. The cursor will not be allowed to move outside the bounds of the file.

The `R` flag invokes the editor in read-only mode, which can be used to view the data but not change it.

6.16 *FFIND fsp (offs) <string/{hex}>

Data in a file can be searched for with this command. The address of each occurrence of the data relative to the start of the file is displayed. The search performed by this command is optimal.

The data to search for is specified by a string or a sequence of hex numbers. The maximum length of the string or sequence of hex numbers is 31 bytes.

An offset can be given to start the search part way through the file. Note that if a sequence of hex numbers is given the offset must be specified, otherwise the first number will be treated as the offset.

6.17 *FORMAT drv <40/80> <X/A> <S/D> (skew)

New discs can be formatted using this command. The drive must be specified, as well as the number of tracks to use, whether the extended **HDFS** catalogue (`X`) or the Acorn catalogue (`A`) are to be used, and whether to format single or double sided (`S` or `D`).

Track numbers are printed as each disc track is formatted.

The skew parameter is a measure of how many sectors are skipped when the drive steps between tracks. The default skew value is 3.

If a disc which is already formatted is in the specified drive, the command will ask for confirmation before formatting.

6.18 *FREE (dsp)

This command lists the available free spaces in the directory specified, or in the current directory if no name is given. The start sector and length in sectors of each free block is displayed, with a summary of the total space available.

6.19 `*INFO afsp`

Information on the files and directories which match an ambiguous file specification is displayed by this command. This information displayed is the name, permissions, load address, execution address, length, and start sector of the file or directory.

6.20 `*LIB (dsp)`

This command sets the current library to the directory name specified. If no directory name is specified, it lists the contents of the current library.

The current library is searched for commands which are `*RUN` if they are not found in the current directory.

6.21 `*READ (drv) addr start (+)end (D)`

This command reads sectors off a disc into memory. Data from the start sector address up to but not including the end sector address will be read into memory, starting at the address given.

The `D` flag reads sectors which have been saved with the 8271 disc controller chip's "deleted data" mode.

The current drive is used if none is given.

6.22 `*RENAME <fsp fsp/afsp dsp> (C)`

Files can be renamed or moved between different directories or discs with this command. There are two different forms for this command; if two filenames are specified, the first file will be renamed to the second. Note that this may involve moving data around if the files are in different directories. If an ambiguous file specification and a directory specification are given, all of the files matching the ambiguous file specification will be moved into the directory. Either a DNFS or an **HDFS** directory can be given.

The `C` flag prompts to change the disc between reading and writing files, so files can be transferred to other discs with one disc drive.

The maximum amount of information is read or written each time the disc is changed, so the fewest number of disc changes possible will be used.

This command overwrites the contents of the I/O processor memory, so it should not be issued if there is valuable data in memory which has not been saved.

6.23 `*TITLE string (dsp)`

The title of a directory can be set with this command. If no directory name is given, the current directory is used. The title is displayed when the directory is listed. The maximum length of a directory title is 12 characters.

6.24 `*TYPE fsp (C)`

This command displays a file as text. All characters are sent to the operating system OSASCII entry, so control characters in the file may cause unexpected (or perhaps, desired) effects, such as changing colour, display mode, *etc*.

The C flag causes non-printing characters to be displayed as their equivalent operating system escape sequence. Newlines are generated after each RETURN (&0D) character.

6.25 *VERIFY drv <40/80> <X/A> <S/D>

This command verifies that the data on a disc is not damaged. The drive number, number of tracks, catalogue type, and number of sides to check are required. The catalogue type is either the **HDFS** extended catalogue (X) or the Acorn (A) catalogue. Single (S) and double sided (D) discs can be checked. The track numbers are printed as the command executes, and a question mark is displayed after any track number on which a verification error occurred.

6.26 *WILD :~^.*?

The special characters which divide file specifications can be altered with this command. Six different characters must be specified, to replace the default characters shown below:

: This is the drive prefix character, which is used before drive specifications in contexts where they are optional.

~ This is the root directory character, which refers to the root directory of the current drive.

^ This is the parent directory character, which is used to refer to the parent directory of the current directory.

. This is the directory separator character, which is used to separate drive specifications from directory specifications, directory names from each other, and directory specifications from file names.

***** This is the glob character, which matches any number of characters in ambiguous file specifications.

? This is the query character, which matches any single character in ambiguous file specifications.

6.27 *WIPE afsp (A)

This command attempts to delete the files specified. It prompts for confirmation of deletion for each unlocked file which matches the file specification. If this command is interrupted, no files will be deleted.

The A flag may be used to confirm all of the files for deletion; this has a similar effect to the *DESTROY command in DNFS.

6.28 *WRITE (drv) addr start (+)end (D)

This command writes sectors to a disc from memory. Data is written at the start sector address up to but not including the end sector address, from the memory address given.

The D flag writes sectors using the 8271 disc controller chip's "deleted data" mode.

The current drive is used if none is given.

7 Utility Commands

These commands are available all the time, even when **HDFS** is not the current filing system.

All of these commands may be prefixed by "`H`" to distinguish them from similar commands in other ROMs.

7.1 `*CLOSE`

This command will close all open files on the current filing system.

7.2 `*DISC`, `*DISK`

These commands have the same action, which is to select the **HDFS** as the current filing system.

7.3 `*GOIO addr`

This command is similar to the second processor command `*GO`, which starts execution at an address in the second processor, except that `*GOIO` starts execution in the I/O processor. It is used from the second processor to run programs downloaded into the I/O processor.

7.4 `*KEYS`

This command lists the current definitions for active function keys, in a form suitable for use with the operating system `*KEY` command.

7.5 `*MEDIT start (+)end ((+)offs) (R)`

This command invokes the disc, file, and memory editor on the section of memory specified by the start address, up to but not including the end address. If an offset is supplied, the cursor will be placed at that point initially.

The `R` flag invokes the editor in read-only mode, which can be used to view the data but not change it.

7.6 *MFIND start (+)end <string/{hex}>

Data in memory can be searched for with this command. The memory address of each occurrence of the data is displayed. The search performed by this command is optimal.

The data to search for is specified by a string or a sequence of hex numbers. The maximum length of the string or sequence of hex numbers is 31 bytes.

The area of the memory to search is delimited by the start and end addresses.

7.7 `*MODE dec`

The display mode is changed by this command. It is equivalent to the BASIC command MODE.

7.8 `*MOVE start (+)end addr`

This command moves a block of memory from one address to another. The block of memory to be moved is delimited by the start and end addresses, and is moved to the other address given. Overlapping source and destination blocks are detected, and handled correctly. The source and/or destination address may be in I/O or second processor memory.

7.9 `*ROMS`

The ROMS installed in the sideways ROM sockets may be listed with this command. The ROM socket number, ROM version number, entry points, and title string are displayed for all sockets. The flags `S`, `L`, and `R` are used to indicate that the ROM has service, language, and/or relocation entries.

7.10 `*VDU <string/{((+/-)dec) (,/;)}>`

This command is used to send a string or sequences of bytes to the operating system OSWRCH entry for display. The syntax of this command may look complicated, but is exactly the same as BASIC's VDU command. A string may be displayed, or a sequence of bytes and words. The numbers in the sequence are positive or negative decimals; a semi-colon after a number indicates that it to be treated as a word (2 bytes). Commas or spaces may be used to separate individual bytes.

The maximum length of string or sequence which can be displayed is 31 bytes.

8 Other commands

The commands in this section are recognised by the operating system, and passed to the filing system through various vectors, primarily OSFILE and FSCV. In some cases, the parameters are interpreted by the operating system; in others, they are passed to the filing system to interpret.

8.1 `*.`

This is shorthand for `*CAT`.

8.2 `*/fsp` ...

This is shorthand for `*RUN`. It allows file specifications including directories to be given easily, without the operating system mistaking the directory for an abbreviated command.

8.3 `*CAT (dsp)`

This command lists the catalogue for the directory specified, or the current directory if none is given. **HDFS** reports the name of the directory being listed, and then the directory title, cycle or key number, number of sectors allocated, start sector, boot-up option, current directory and current library before the actual catalogue.

The catalogue is listed in alphabetical order, with the permissions after each file. Acorn format (DNFS) catalogues are listed with the current directory first and its' directory character missing, and then the other directories, in alphabetical order. In **HDFS** format catalogues, the flag `D` next to the permissions indicates that the entry is a directory.

In 80-column modes, four files are displayed on each line; in 20 and 40-column modes, two files are listed on each line.

8.4 `*EXEC fsp`

Files can be used as keyboard input with this command. The contents of the file are operated on exactly as if they had been typed at the keyboard. The main use of `*EXEC` is for auto-boot files which run applications.

8.5 `*HELP (HDFS/UTILS)`

This command prints out the help available from the ROM. With no keyword, **HDFS** prints the version number, and the keywords to which `*HELP` will respond.

If the `HDFS` keyword is given, the syntax of all of the filing system commands in section 6 will be printed.

The `UTILS` keyword causes the syntax of all of the utility commands in section 7 to be printed.

8.6 `*LOAD fsp (addr)`

This command loads a file to the address specified, or to its' own load address if no address is given. The address may be in I/O processor memory or second processor memory; if the address is in I/O processor memory, and it would erase the **HDFS** private workspace, the ROM filing system (RFS) is invoked after the file is loaded. This allows large files to be loaded to low memory addresses safely. If the address is in I/O processor memory, and in the range &8000–&BFFF, the data is loaded into the sideways ROM socket given by the value of `*OPT 3`.

8.7 `*OPT dec(,dec)`

This command sets the filing system options. The numbers determine which option is to be set, and what its value is to be. Section 11.7 contains a list of what effects the options have in **HDFS**.

8.8 `*RUN fsp` ...

This command loads and executes a file. The file is loaded to its' own load address, and executed at its' own execution address. The rest of the command line to `*RUN` is made available for the program to read through the OSARGS call.

The file is first searched for in the current directory, and then in the current library if that fails.

In **HDFS**, files can only be executed if they have execute permission set. Files on Acorn format catalogues can always be executed.

As a special case, if the execution address is &FFFFFFFF, the file will be *EXECed instead of run. If a directory is *RUN, it will be made the current directory, and its' auto-boot options will be used to determine whether to look for a !BOOT file to load, run or *EXEC.

8.9 *SAVE fsp start (+)end (exec (load))

Blocks of memory can be saved to files with this command. The memory to be saved is delimited by the hexadecimal start and end addresses. The execution address and load address of the file saved can be set by the optional exec and load arguments.

If the addresses to be saved are in I/O processor memory, and in the range &8000–&BFFF, the data is saved from the sideways ROM socket given by the value of *OPT 3.

8.10 *SPOOL (fsp)

This command sends output to the screen to a disc file as well. It can be used to get a textual representation of almost anything that can be printed into a file.

If no file specification is given, *SPOOLed output is turned off.

In **HDFS**, filing system commands which produce output turn off *SPOOLed output, because of potential race conditions.

9 The Disc, File, and Memory Editors

The *DEDIT, *FEDIT, and *MEDIT commands invoke an interactive hex and ASCII data editor. The user interface is the same for all of these commands.

All of the editing commands can be invoked in read-only mode, which allows viewing but not alteration of the data.

The display format for the editor is similar to *DUMP; on each row, the address of the data is displayed at the left, followed by a number of bytes in hexadecimal representation, and then the same bytes in ASCII representation. The number of bytes which are displayed on each row is 16 in 80 column display modes, and 8 in 40 column modes. The editor cannot be used in 20 column modes.

In the file editor, the address displayed is the offset from the start of the file. The disc editor displays the absolute sector address, and the memory editor displays the absolute memory address.

The cursor can be moved about using the cursor keys. The left and right keys move one byte forwards or backwards, wrapping around to the next line at the edges of the screen. The up and down keys move forward or backward by one row. Larger movements up and down are possible using SHIFT or CTRL with the cursor keys. SHIFT with the up and down keys moves the cursor one screenfull forward or backwards. CTRL with the up and down keys moves the cursor to the start or end of the data. The cursor can be moved to the start or end of the current line by using CTRL with the left or right keys. The RETURN key moves the cursor to the start of the next line.

The editor starts with the cursor in the hexadecimal data area. It can be moved to the ASCII data area by pressing SHIFT with the right cursor key, and back to the hexadecimal area with SHIFT and the left cursor key. When in the hexadecimal area, typing hexadecimal symbols will alter the current byte. In the ASCII area, the code for the character typed is inserted, and the cursor is moved one place forward. If the editor is in read-only mode, no alteration is made.

The editor has a buffer of 512 bytes of data which may be modified before it is updated to disc or memory. The buffer is updated automatically if you move outside the area which it holds, and it may be copied back at any time by pressing the COPY key. If an attempt is made to quit the editor while the buffer has not been updated, the editor will sound the bell, and continue normally. If a second attempt is made to quit immediately, the editor will then exit.

The ESCAPE key is used to quit the editor.

A summary of the keys recognised is given below:

Key	Normal	SHIFT	CTRL
COPY	Update to disc/memory		
ESCAPE	Quit editor		
RETURN	Start of next line		
←	Backward byte	Hex area	Start of row
→	Forward byte	ASCII area	End of row
↑	Back row	Back screen	Start of data
↓	Forward row	Forward screen	End of data

10 OSWORD

The operating system's OSWORD entry point is used to provide some functions which are of general use.

On entry to OSWORD, the accumulator contains a value which determines the operation to be performed. X and Y contain the address of a parameter block which is used to pass parameters to and from OSWORD (X low byte, Y high byte). The parameter block may reside in either the I/O or second processor memory.

The OSWORD codes recognised by **HDFS** are listed in the following sections.

10.1 A=&7B Move a block of memory

This call has the same effect as the `*MOVE` command. A block of memory is transferred from one memory location to another. The source and destination addresses can be in I/O or second processor memory.

The format of the parameter block is:

00 01 02 03	Source address of data
04 05 06 07	End address of data
08 09 0A 0B	Destination address

10.2 A=&7C Multiple track 8271 command

This call applies an 8271 disc controller command to a range of sectors on the disc. The 8271 command used determines whether data is read from or written to the disc. The command code used should be one of the 8271 multiple sector commands; appendix D lists the 8271 commands.

The parameter block format is:

00	Drive number
01 02 03 04	Source or destination data address
05 06	Start sector
07 08	Number of sectors to transfer
09	8271 command

If the drive number is &FF, the current drive will be used.

10.3 A=&7D Read cycle number

The disc cycle number or key can be read with this call. The disc cycle number is a number which is used to distinguish discs with similar catalogues. It is normally the number of times the catalogue on the disc has been written, but on **HDFS** root directories, it is a one-byte checksum of the directory.

The format of the result block is:

00	Cycle number

10.4 A=&7E Read directory size

The size of the current directory (in bytes) can be read with this call.

The format of the result block is:

00 01 02 03	Directory size

10.5 A=&7F Single track 8271 command

This call executes a single 8271 disc controller command, and returns the result code. The number of parameters required depends on the 8271 command used. The 8271 command used determines whether data is read from or written to the disc; appendix D lists the 8271 commands. The result code is written into the next location after the parameters.

The format of the parameter block is:

00	Drive number
01 02 03 04	Source or destination data address
05	Number of parameters to 8271 command, n
06	8271 command
07 ⋮ $7+n$	8271 command parameters
$8+n$	Space for result code

If the drive number is &FF, the current drive will be used.

11 Vectors

Filing systems must provide a series of seven vectors when selected. These vectors point to relevant routines within the filing system.

The filing system vectors are:

&212	FILEV	Operations on whole files
&214	ARGSV	Read/write file arguments
&216	BGETV	Get one byte from an open file
&218	BPUTV	Put one byte to an open file
&21A	GBPBV	Get/put a block of bytes to/from an open file
&21C	FINDV	Open/close a file for byte access
&21E	FSCV	Various filing system control actions

The **HDFS** supports all of the filing system vectored operations, with a few restrictions. Some extended operations are also provided; these are marked *.

All of the addresses and file lengths in the parameter blocks are specified least significant byte first.

11.1 OSFILE

Call address &FFDD, indirected through &212.

On entry, X and Y point to a parameter block in memory (X low byte, Y high byte). The format of the parameter block is:

00 01	Address of the filename, terminated by RETURN (&0D)
02 03 04 05	Load address of the file
06 07 08 09	Execution address of the file
0A 0B 0C 0D	Start address of data for save, length of file otherwise
0E 0F 10 11	End address of data for save, file attributes otherwise

The accumulator contains a number indicating the action to be performed:

A=&FF Load the named file at the address given in the parameter block if the least significant byte of the execution address (XY+6) is zero. If this byte is not zero, load the named file at its own load address.

A=0 Save a block of memory using the name and addresses given in the parameter block.

A=1 Change the attributes, load and execution addresses of the named (existing) file.

A=2 Change the load address of the named file only.

A=3 Change the execution address of the named file only.

A=4 Change the attributes of the named file only.

A=5 Read the attributes, load and execution addresses, and length of the named file into the parameter block.

A=6 Delete the named file.

A=7* Create a catalogue entry for the named (non-existent) file, with the attributes, load and execution addresses, and length supplied. A check is made to ensure that the length does not exceed the space available for the file. No data is written to the file.

A=8* Create a new directory with the given name, attributes, addresses, and length supplied. The load and execution addresses are written to the catalogue, but are not actually used for anything. A check is made to ensure that the length does not exceed the space available for the file. A new catalogue is initialised in the directory.

A=9* Change the length of the named (existing) file. A check is made to ensure that the new length does not exceed the space available for the file.

A=10* Change the attributes, load and execution addresses, and length of the named file. A check is made to ensure that the new length does not exceed the space available for the file.

Only the least significant byte of the attributes is used, for the file permissions. There are two bits for each attribute; if either of the bits are set, the attribute is set.

Bit	Bit	Meaning
0	4	Not readable
1	5	Not writable
2	6	Not executable
3	7	Not deletable

On exit, the file type is returned in the accumulator:

0 Nothing found
1 File found
2 Directory found

X and Y are preserved. C, N, V, and Z are undefined.

11.2 OSARGS

Call address &FFDA, indirected through &214.

On entry, X points to a four byte zero page control block. Y contains a file handle (as provided by OSFIND), or zero. The accumulator contains a number specifying the action required:

Y=0 A=&FF All open files are updated to disc if necessary.

A=0 The current filing system number is returned in the accumulator. The filing system number for **HDFS** is 4.

A=1 The address of the rest of the command line is returned in the zero page control block, giving access to parameters passed by `*RUN` or `*`*`command`*. The command line is always in the I/O processor memory.

Y≠0 A=&FF Update the file to disc if necessary.

A=0 Read the sequential pointer of the file into the zero page control block.

A=1	Write the sequential pointer of the file from the zero page control block. If the file is opened for writing or update, and the new pointer is greater than the length for the file, the file is extended and filled with zero data bytes to the required length.
A=2	Read the length of the file into the zero page control block.
A=3*	Read the maximum length that the file can reach into the zero page control block. It may be possible to make more space for the file by shuffling other files around.

On exit, X and Y are preserved. The accumulator is preserved, except when reading the filing system type. C, N, V, and Z are undefined, and D is cleared.

11.3 OSBGET

Call address &FFD7, indirected through &216.

On entry, Y contains the file handle, as provided by OSFIND.

A byte is read from the point in the file designated by the sequential file pointer.

On exit, X and Y are preserved. The accumulator contains the byte read. C is set if the end of file has been reached, and indicates that the byte obtained is invalid. N, V, and Z are undefined.

11.4 OSBPUT

Call address &FFD4, indirected through &218.

On entry, Y contains the file handle, as provided by OSFIND. The accumulator contains the byte to be written to the file.

The byte is written at the point in the file designated by the sequential file pointer.

On exit, X, Y, and A are preserved. C, N, V, and Z are undefined.

11.5 OSGBPB

Call address &FFD1, indirected through &21A.

On entry, X and Y point to a control block in memory (X low byte, Y high byte). The control block format is:

00	File handle
01 02 03 04	Address of the data
05 06 07 08	Number of bytes to transfer
09 0A 0B 0C	New sequential pointer to be used for transferring data

The accumulator contains a number indicating the action to be performed:

A=1 Put bytes to disc, using the new sequential pointer

A=2 Put bytes to disc, ignoring the new sequential pointer

A=3 Get bytes from disc, using the new sequential pointer

A=4 Get bytes from disc, ignoring the new sequential pointer

A=5 Get the disc title and boot up option. The data returned is in the format:

00	Length of the title, n
01 ⋮ n	Disc title
$n+1$	Boot up option

The boot option is the value that was set with `*OPT` or FSCV (see section 11.7).

A=6 Read the current directory and device. The data returned is in the format:

00	1
01	Drive number
02	Length of current directory name, n
03 ⋮ $n+3$	Current directory name

A=7 Read the current library name and device. The data is returned in the same format as the current directory (A=6).

A=8 Read file names from the current directory. The control block is modified, so that the file handle byte contains the cycle number or directory key, and the sequential pointer is adjusted so that the next call with A=8 will get the next file name. The number of bytes to be transferred is interpreted as the number of file names to read; for the first call, the sequential pointer should be zero. The data is returned in the format:

00	Length of filename 1, n_1
01 ⋮ n_1	Filename 1
n_1+1	Length of filename 2, n_2
n_1+2 ⋮ $n_1+\cdots$	Filename 2
$\cdots$	*etc.*

A=9* Get spaces from the current directory. The data returned is in the format:

00 01	Start sector of gap 1
02 03	Length in sectors of gap 1
04 05	Start sector of gap 2
06 07	Length in sectors of gap 2
···	*etc.*

On exit, X, Y, and the accumulator are preserved. N, V, and Z are undefined. The C flag set if the end of the file has been reached, or if there are no more file names or gaps to read. In this case, the number of bytes, names or gaps which have not been transferred are written back to the parameter block. The address field and sequential pointer are always adjusted to point to the next byte to be transferred.

11.6 OSFIND

Call address &FFCE, indirected through &21C.

On entry, the accumulator specifies what action is to be performed:

A=0 A file is to be closed:

- Y=0 Close all files
- Y≠0 Y contains the file handle of the file to close

A≠0 A file is to be opened. X and Y point to the file name (X low byte, Y high byte). The file name is terminated by RETURN (&0D). The accumulator indicates what type of access is required:

- A=&40 The file is to be opened for input only
- A=&80 The file is to be opened for output only. If a file with the same name exists, an attempt is made to delete it before opening for output.
- A=&C0 The file is to be opened for update (input and output). The file will not be created if it does not exist.

On exit, X and Y are preserved. The accumulator is preserved on closing, and on opening contains the file handle assigned to the file. If the accumulator is zero on exit, the file could not be opened. C, N, V, and Z are undefined.

11.7 FSCV

There is no direct address to FSCV, indirect access is through &21E.

On entry, the accumulator contains a number specifying what action is to be performed:

A=0 Perform a `*OPT` command; X and Y are the two parameters. The value of X determines what action is taken:

X=0 Restore default values. The defaults values are:

X	Value	Meaning
1	0	No filing system messages
2	5	Number of retries
3	&F	Sideways ROM socket

X=1 Turn on filing system messages. The value of Y determines the amount of information given:

- Y=0 Turn off filing system messages
- Y=1 File name and permissions are displayed
- Y=2 File name, addresses, length and permissions are displayed

If filing system messages are turned on, they are displayed on most OSFILE and OSFIND operations, and by some other commands as well.

X=2 The value of Y is the number of retries made before giving up when a disc error is detected.

X=3 The value of Y is the ROM socket to be used when loading and saving to I/O processor addresses in the range &8000–&BFFF.

X=4 The auto-boot option of the current directory is set according to the value of Y:

- Y=0 No action
- Y=1 `*LOAD !BOOT`
- Y=2 `*RUN !BOOT`
- Y=3 `*EXEC !BOOT`

A=1 Check whether end of file (EOF) has been reached. On entry X is the file handle to be checked. On exit, X is &FF if EOF has been reached, and zero if EOF has not been reached.

A=2 A `*/` command has been used. The command whose name follows the '/' character will be `*RUN`.

A=3 An unrecognised operating system command has been used. **HDFS** will check the command against its own filing system commands first, and `*RUN` the command if it is not found. On entry X and Y point to the command line (X low byte, Y high byte).

A=4 A `*RUN` command has been issued. The file name pointed to by X and Y (X low byte, Y high byte) will be loaded and executed. As a special case, if the execution address of the file is &FFFFFFFF, the file will be `*EXEC`ed instead.

A=5 A `*CAT` command has been used. X and Y point to the rest of the command line.

A=6 Shut down **HDFS** because a new filing system is taking over.

A=7 The lowest and highest possible file handles used are returned in X and Y. **HDFS** uses the range &12–&17.

A=8 This call is used whenever an operating system command is about to be processed. It is used by DNFS to implement the `*ENABLE` flag. **HDFS** takes no action on this entry.

12 Errors

A wide variety of errors can be produced by the **HDFS**. All of the errors use the BBC's normal error reporting mechanism, and so can be trapped within programs easily. The errors are described in the following sections. The error numbers are given in the section headings, and where possible are the same as DNFS errors.

12.1 Escape (&11)

The `ESCAPE` key was pressed to abort an operation. This can only happen at certain points; it is usually disabled during data transfer.

12.2 Bad end (&B0)

This error occurs the end sector address given for a command is too large, or when the end sector or memory address given is smaller than the corresponding start address. It can also occur if a block of memory specified by start and end addressed crosses over from the second processor space to the I/O processor space.

12.3 Bad start (&B1)

This error occurs the start sector address given for a command is too large, or when the offset specified to the `*FEDIT` command is larger than the file length.

12.4 Bad length (&B2)

An attempt was made to create a directory with an invalid length, or modify the length of a file or directory to a size too big for the available space.

12.5 Bad range (&B3)

An offset was specified to a command which was not between the start and end addresses given.

12.6 Not empty (&B4)

An attempt was made to delete a directory that was not empty. The contents of the directory should be deleted first.

12.7 Too long (&B5)

This error occurs when a string or sequence of hex numbers exceeds the maximum length allowed.

12.8 Bad wild char (&B6)

This error is produced when the command `*WILD` is used to try to set wild card characters which are not all different, printable characters.

12.9 Find what? (&B7)

An empty string was given to the `*DFIND`, `*FFIND`, or `*MFIND` commands.

12.10 Bad edit mode (&B8)

The current display mode is not capable of supporting the disc, file and memory editor.

12.11 Parameters (&B9)

Too many parameters were provided for an 8271 controller command.

12.12 Wrong format (&BA)

This error indicates that an attempt was made to access a disc which was not in a valid format. This usually occurs when the reverse side of a double sided **HDFS** format disc is accessed with drive numbers `:2` or `:3`.

12.13 Not readable (&BB)

An attempt was made to open a file for input which is not marked as readable.

12.14 Not executable (&BC)

A command file was `*RUN` which is not marked as executable.

12.15 Too big (&BD)

The maximum size of number accepted by any **HDFS** command is &FFFFFFFF for hexadecimal numbers, or 65535 for decimal numbers. This error occurs when a number larger than these limits is specified.

12.16 Cat full (&BE)

There are already 31 entries in the directory catalogue. Partitioning files into sub-directories can free up some more catalogue space.

12.17 Can't extend (&BF)

A file open for output or update has run out of the available space. This error should occur less frequently in **HDFS** than DNFS, because new files are opened in the largest gap available in a directory.

12.18 Too many open (&C0)

Too many files were opened simultaneously; the maximum number of files which **HDFS** can have open at one time is 6.

12.19 Not writable (&C1)

An attempt was made to open a file which is not marked as writable for update or output.

12.20 Open (&C2)

An attempt was made to open a file which was already open, or an operation was required which cannot be performed on open files. Operations which move file contents cannot be performed on open files.

12.21 Locked (&C3)

The file or directory is locked, and the operation required cannot be performed without unlocking it first.

12.22 Exists (&C4)

This error is issued when a file or directory specification which exist was specified in a context in which a new name was required.

12.23 Drive fault EE at DD/TT (&C5)

This error indicates that an error was detected with the disc drive, and that a transfer could not be performed. The error code, drive number and track address are given in the error message.

The error code given indicates what the error was:

`0A`	Late DMA
`10`	Drive not ready
`14`	Track 0 not found
`16`	Write fault

These errors are probably unrecoverable.

The first digit of the drive number indicates which disc surface was being accessed when the fault occurred. This may not be the same as the drive number for **HDFS** format discs, which can span the front and back sides of the same disc. The surface codes are:

4 Surface 0 (side 1 of drive 1)
8 Surface 1 (side 1 of drive 2)
6 Surface 2 (side 2 of drive 1)
A Surface 3 (side 2 of drive 2)

12.24 Dir full (&C6)

No more space is available in the directory. If it is not the root directory, the directory may be able to be `*EXTEND`ed, and given a larger length with `*ATTRIB`.

12.25 Disc fault EE at DD/TT (&C7)

This error indicates that an error was detected on the disc, and that a transfer could not be performed. The error code, drive number and track address are given in the error message.

The error code given indicates what the error was:

08 Clock error
0C ID CRC error
0E Data CRC error
18 Sector not found

These errors are probably worth trying again; the maximum number of re-tries can be set using `*OPT 2`.

The first digit of the drive number indicates which disc surface was being accessed when the fault occurred. This may not be the same as the drive number for **HDFS** format discs, which can span the front and back sides of the same disc. The same codes as the drive fault error are used.

12.26 Disc changed (&C8)

The disc on which data should have been written has been changed. This can happen when files are opened and the disc is changed, or when the wrong disc is inserted while copying files with one disc drive.

12.27 Disc read only (&C9)

An attempt was made to write to a disc which has its write-protect notch covered.

12.28 Bad track (&CA)

A track number which is larger than the current disc can support has been used. This error sometimes occurs when **HDFS** double-sided discs are used with a valid track number. In these cases, specifying the drive number to the command should make it work correctly.

This can also occur when copy-protection schemes are used; an option to the `*DISC` command is available to turn off track number checking for these discs.

12.29 Bad option (&CB)

This error occurs when a `*OPT` command is used with the code or value out of range. See section 11.7 for the meaning of the `*OPT` parameters.

12.30 Bad name (&CC)

An invalid file or directory name has been used. Possible causes for an invalid name are; wild card characters were used in the directory part, the name is too long, or a directory or file name part is longer than 7 characters.

12.31 Bad drive (&CD)

A drive number outside the range 0–3 was used.

12.32 Directory (&CE)

A directory specification was given in a context in which only files can be used.

12.33 Not found (&D6)

This error is issued when a file or directory specification which does not exist was used, in a context in which it was required.

12.34 Syntax: ... (&DC)

Incorrect command syntax has been used. The correct syntax for the command is shown in the error message.

12.35 Channel (&DE)

An invalid file handle was used; this may be because the file had been already closed, or had not been opened successfully.

12.36 EOF (&DF)

An attempt was made to read past the end of a file. An error flag is set on the first attempt to read past end of file (see section 11.3), and this error is issued on the next attempt to read.

12.37 Bad command (&FE)

An unrecognised operating system command was issued, and it could not be found in the current directory or library. This error is also issued by attempts to `*RUN` non-existent files.

13 Compatibility

The **HDFS** is compatible with DNFS, but not identical. Most of the commands are the same, but there are some differences. These are:

Unsupported commands. The following commands are not supported: `*BACKUP`, `*DESTROY`, and `*LIST`. `*BACKUP` is replaced by the new syntax for `*COPY`, and `*DESTROY` is replaced by the option to `*WIPE`. `*LIST` was not used much, and is not replaced.

Abbreviations. The differences in the filing system commands means that some abbreviations which work with DNFS (*e.g.*, `*DE.`) will not work with **HDFS**. The safe method is to use full command names.

Utility commands. The DNFS utility commands `*BUILD`, `*DUMP`, and `*TYPE` are filing system commands in **HDFS**, which means that they can only be used when **HDFS** is active. The line length on `*BUILD` is limited to 63 characters.

Command syntax. There are a few syntax changes. `*CAT` and `*COMPACT` take a directory as an argument instead of a drive. The syntax for `*COPY` and `*RENAME` has changed, so that they take two file specifications or an ambiguous file specification and a directory specification instead of two drive numbers and an ambiguous file specification.

A space is needed after all **HDFS** commands, where it was not necessary in DNFS *e.g.*, `*DIR :0` instead of `*DIR:0`.

File name syntax. The single character match wildcard in **HDFS** is normally "?", instead of the "#" used for DNFS. It can be changed with the `*WILD` command if necessary. The "~" and "^" characters are also special in **HDFS**.

No enabling. The `*ENABLE` command is provided, but is not required by **HDFS**.

Spooled output. **HDFS** is not re-entrant, and so it turns off output to the `*SPOOL` file in certain circumstances which could cause race conditions. Most DNFS output can be spooled.

Catalogue reporting. The format of the catalogue displayed by **HDFS** is slightly different than DNFS. The current directory and library take more space, and the directory start sector has been added to the information. File permissions are reported differently, because **HDFS** supports permissions for reading, writing, and executing files.

Display modes. **HDFS** makes full use of the 80 column modes. Catalogues and data dumps use the full width of the screen, whereas DNFS just produces 40 column output whatever the display mode.

The same memory areas are used by **HDFS** and DNFS; see appendix F for details of the **HDFS** memory usage.

Some copy protection schemes which were designed to run under DNFS may not initially work with DNFS, because of the auto-stepping and sector number checking features. The extra parameters to the `*DRIVE` command can be used to disable sector checking, and set the step rate to single or double stepping to allow these schemes to work. The OSWORD &7F command for access to the 8271 disc controller chip is fully supported by the **HDFS**.

A Overview

Floppy discs are flat discs made from a similar material to cassette tape, and are magnetised to store information.

Floppy discs and disc drives come in many different sizes and formats. The most popular sizes now are 5.25 inch and 3.5 inch, but 8 inch discs are still available. There are three things which determine how much information a disc can store; the number of tracks supported by the disc drive, whether the disc drive is single or double sided, and the recording density. Information is written on to floppy discs in a series of concentric bands, or tracks. Most disc drives will be able to read and write 80 of these tracks, but some older drives may only support 40 tracks. An 80 track disc will store twice as much information as a 40 track disc. Double sided disc drives have two read/write heads, one on each side of the disc, and so are able to record twice as much information as a single sided drive, which has only one read/write head. Note: single sided discs cannot be inverted to use the reverse side of the disc, except for 3.5 inch discs. The recording density determines how much information can be stored on each track of a disc. Single density recording allows 10 *sectors* of 256 bytes each to be recorded on a typical disc, whereas double density recording allows up to 18 sectors per track. The Intel 8271 Floppy Disc Controller chip which **HDFS** is designed to work with can only read and write single density discs.

HDFS is designed to work with single or double sided, 40 or 80 track disc drives. The amount of information which can be stored on a disc is given in the table below.

	Tracks	
Sides	40	80
Single sided	100k	200k
Double sided	200k	400k

B Installation

When you have inserted the **HDFS** ROM and turned the machine on, you should see a message like:

```
BBC Computer 32K

Hierarchical DFS

BASIC

>
```

This message indicates that the **HDFS** is selected as the current filing system. It is possible to have the DNFS installed as well as the **HDFS**. In this case, the ROM in the higher priority socket will take control by default. If the **HDFS** is in the lower priority socket, it may be selected by the key combination H-BREAK. The DNFS may be selected by pressing D-BREAK.

The **HDFS** also allows commands to be prefixed by “`H`”, so it can also be selected unambiguously by the command `*HDISC`.

There are several link switches at the bottom right hand corner of the keyboard PCB; some of these can be used to set options to **HDFS**. Links 3 and 4 are used to set the disc drive characteristics, and links 1 and 2 are used to set the default stepping configuration.

The disc drive timing link positions are:

Link 3	Link 4	Step time	Settle time	Head load
1	1	4	16	0
1	0	6	16	0
0	1	6	50	32
0	0	24	20	64

The drive stepping configuration links are:

Link 1	Link 2	Action
1	0	Double stepping enabled
0	1	Single stepping enabled
0	0	Automatic double stepping enabled

C Command Summary

This section contains a list of the **HDFS** commands and their syntax.

```
ACCESS afsp (LXWR)
ATTRIB fsp (load exec) (len) (LXWR)
BUILD fsp (A)
CLOSE
COMPACT (dsp) (D)
COPY <fsp fsp/afsp dsp> (C)
CREATE fsp (load exec) (len) (LXWR) (D)
DEDIT (drv) start (+)end ((+)offs) (R)
DELETE fsp
DFIND (drv) start (+)end <string/{hex}>
DIR (dsp)
DISC
DISK
DRIVE drv (S/D/A) (C)
DUMP fsp
ENABLE
EXTEND fsp
FEDIT fsp (offs) (R)
FFIND fsp (offs) <string/{hex}>
FORMAT DRV <40/80> <X/A> <S/D> (skew)
FREE (dsp)
GOIO addr
INFO afsp
KEYS
LIB (dsp)
```

```
MEDIT start (+)end ((+)offs) (R)
MFIND start (+)end <string/{hex}>
MODE dec
MOVE start (+)end addr
READ (drv) addr start (+)end (D)
RENAME <fsp fsp/afsp dsp> (C)
ROMS
TITLE string (dsp)
TYPE fsp (C)
VDU <string/{((+/-)dec) (,/;)}>
VERIFY drv <40/80> <X/A> <S/D>
WILD :~^.*?
WIPE afsp (A)
WRITE (drv) addr start (+)end (D)
```

D 8271 commands

This section describes the operations supported by the 8271 floppy disc controller chip. These commands can be sent to the 8271 using the OSWORD &7F call. Refer to the 8271 FDC data sheet for more information.

Some of the commands in the table below operate on multiple sectors; for these commands, the sectors must all be on the same track. Other data transfer commands operate on single sectors only. The commands supported by the 8271 FDC are:

Command	Op	P1	P2	P3	P4	P5
Scan multi	00	Track	Sector	Len/Sec	Scan step	Field length
Scan deleted multi	04	Track	Sector	Len/Sec	Scan step	Field length
Write	0A	Track	Sector			
Write multi	0B	Track	Sector	Len/Sec		
Write deleted	0E	Track	Sector			
Write deleted multi	0F	Track	Sector	Len/Sec		
Read	12	Track	Sector			
Read multi	13	Track	Sector	Len/Sec		
Read deleted	16	Track	Sector			
Read deleted multi	17	Track	Sector	Len/Sec		
Read track ID	1B	Track	0	NumID		
Verify deleted	1E	Track	Sector			
Verify deleted multi	1F	Track	Sector	Len/Sec		
Format multi	23	Track	Gap 3	Len/Sec	Gap 5	Gap 1
Seek	29	Track				
Read status	2C					
Specify	35	What	Step/ Track	Settle/ Track	Load/ Track	
Write register	3A	Register	Data			
Read register	3D	Register				

The parameters for these commands, and their meanings are:

Track This is the track number (normal range 0-79).

Sector This is the start sector on the track (normal range 0-9).

Len/Sec Bits 0–4 indicate the number of sectors accessed (0–9). Bits 5–7 are sector size (2 for 256 byte sectors).

NumID This is the number of ID fields to transfer.

Scan step Bits 6–7 indicate the scan condition (0 EQ, 1 GEQ, 2 LEQ). Bits 0–5 are the offset to the next sector in a multi scan.

Field length This is the number of bytes to compare when scanning.

Gap 1 This is the formatting gap between the last sector and the index mark.

Gap 3 This is the syncronisation field formatting gap.

Gap 5 This is the formatting gap between the ID field and sector.

What This value indicates what the other parameters in the command mean. The values &10 and &18 load surface zero and one bad tracks, and &0D sets the drive parameters.

Step This is the disc step rate.

Settle This is the disc head settle time.

Load Bits 0–3 are the head load time, Bits 4–7 are the index unload count.

Register This is the register number to read or write.

Data This is the data to read or write to a register.

The result byte returned by the 8271 contains the result value of the command. Bit 5 of the result is a flag which indicates that deleted data was found on the disc. Bits 1–4 contain a completion code which indicates what the final status of the command was:

`00`	Good completion or scan not met
`02`	Scan met equal
`04`	Scan met not equal
`08`	Clock error
`0A`	Late DMA
`0C`	ID CRC error
`0E`	Data CRC error
`10`	Drive not ready
`12`	Write protect
`14`	Track 0 not found
`16`	Write fault
`18`	Sector not found

Note that the scan commands do not behave as specified in the 8271 data sheet. The memory address of the scan key is not automatically reset each time the scan repeats, and so the scan key should be duplicated throughout a memory buffer the same size as the number of sectors being scanned. This behaviour is the same under DNFS and **HDFS**.

E Catalogue Format

The format of **HDFS** and Acorn (DNFS) catalogues differs slightly. The catalogue formats are similar enough that **HDFS** discs can be catalogued using DNFS, but no changes should be made to them. There are two sectors allocated to each catalogue:

Sector 0:

	Bytes	Data	DNFS	**HDFS**
	0	Title char 1		b7: Sectors bit 11
	1–7	Title chars 2–8		
	8	Filename char 1		b7: Start sec bit 11
	9	Filename char 2		b7: Length bit 18
	A	Filename char 3		
31 ×	B	Filename char 4		b7: Directory
	C	Filename char 5		b7: Not readable
	D	Filename char 6		b7: Not writable
	E	Filename char 7		b7: Not executable
	F	b7: Not deletable	b0–6: Directory char	b0–6: 0

Sector 1:

	Bytes	Data	DNFS	**HDFS**
	0–3	Title chars 9–12		
	4		Cycle number	Key number
	5	Entries × 8		
	6	b0–1: Sectors bits 8–9		
			b2: 0	b2: Disc sides 1/2
			b3: 0	b3: 1
		b4–5: Auto-boot option		
		b6–7: 0		
	7	Sectors bits 0–7		
	8–9	Load address bits 0–15		
	A–B	Exec address bits 0–15		
	C–D	Length bits 0–15		
31 ×	E	b0–1: Start sec bits 8–9		
		b2–3: Load bits 16–17		
		b4–5: Length bits 16–17		
		b6–7: Exec bits 16–17		
	F	Start sector bits 0–7		

F Memory Usage

F.1 Zero page

Several zero page areas are used by **HDFS**.

The NMI workspace from &A0–&A7 is used when the NMI area is claimed for disc transfers.

The operating system scratch workspace from &AC–&AF is used for parsing commands.

The filing system scratch workspace from &B0–&BF is used as the main zero page workspace. The contents are used during **HDFS** command execution, but are not held over between commands.

The zero page filing system workspace from &C0–&CF is fully used while **HDFS** is active. This area should not be changed by user programs:

C0–C3	Drive information table
C4–C5	Directory start sector
C6	Copy of current drive
C7	Last drive accessed
C8–C9	Command line parameters
CA	Last catalogue type
CB	Root key of last drive
CC	Tube flag
CD	Message flag (`*OPT 1` value)
CE	Number of retries (`*OPT 2` value)
CF	Sideways ROM number (`*OPT 3` value)

F.2 NMI workspace

The NMI workspace in &D00–&D9E is claimed for disc and memory transfers. The NMI allocation regime is non-greedy; the workspace is returned to its previous owner as soon as possible. The zero page NMI workspace is also used when the NMI workspace is claimed.

F.3 Absolute workspace

The absolute workspace in &E00–&11FF is claimed for use by most operations. **HDFS** is greedy about absolute workspace; it will not release control of the absolute workspace until requested by another claimant.

Absolute workspace addresses &E00–&FFF are used to hold a copy of the current catalogue. This copy is used instead of reading the catalogue off the disc if the disc is still spinning when an operation is started.

The absolute workspace in &1000–&11FF is used for various purposes:

1000–101F	Scratch filename 1
101F–103F	Scratch filename 2
1040–105F	Current directory
1060–107F	Current library
1080–1083	Copy of drive information table
1084–1089	Character table
108A–108F	File handle table
1090–109F	Command workspace
10A0–10DF	String space
10E0–10F5	File info workspace
1100–1115	Open file info 1
1116–112B	Open file info 2
112C–1141	Open file info 3
1142–1157	Open file info 4
1158–116D	Open file info 5
116E–1183	Open file info 6
11A0–11AF	OSGBPB workspace
11B3–11FF	Open file workspace

F.4 Private workspace

Seven pages of private workspace are allocated; these will usually be from &1200–&17FF. The first six pages are used to keep memory buffers for open files. The last page of private workspace, which is usually in addresses &1800–&18FF, is used for a copy of vital information which is normally kept in the absolute workspace &1000–&11FF. This is used when the filing system is shut down and re-activated. This memory page contains:

00–1F	Current directory
20–3F	Current library
40–43	Drive table
44–49	Character table
4A–4F	File handle table
50–65	Open file info 1
66–7B	Open file info 2
7C–91	Open file info 3
92–A7	Open file info 4
A8–BD	Open file info 5
BE–D3	Open file info 6
D4	Private workspace validity flag

G Benchmarks

The benchmarks in the tables below indicate how the performance of **HDFS** compares to DNFS. There are four columns in each table; the first three columns are results for **HDFS** operating on a native **HDFS** root directory, a single level sub-directory, and an Acorn format directory. The final column contains results for Acorn's DNFS 1.20. All of the times shown are in seconds, most of them timed over 10 repetitions.

All of the tests were performed on discs formatted with a skew of 3.

G.1 SAVE 32K

In this test, &8000 bytes were saved to a file. Several variants of this test were performed. In the tables below, "IO" indicates that the data was taken from the I/O processor, and "Tube" indicates that the data was taken from the second processor. The keyword "Spun" indicates that the disc was spinning when the test started, so the drive start-up time is not counted in the timing. The word "Stop" indicates that the disc was stopped when the test was started, and the word "New" indicates that a new file entry was used, rather than writing over an old file.

SAVE 32K	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
IO, Spun	4.52	4.66	3.83	3.84
IO, Stop	4.82	5.06	4.43	4.40
Tube, Spun	4.54	4.62	3.83	3.87
Tube, Stop	4.73	5.17	4.42	4.49
IO, Spun, New	4.39	4.81	3.84	3.72
IO, Stop, New	4.73	5.04	4.43	4.33
Tube, Spun, New	4.41	4.83	3.84	3.75
Tube, Stop, New	4.85	5.11	4.44	4.37

G.2 LOAD 32K

This test measured the time taken to load &8000 bytes from a file.

LOAD 32K	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
IO, Spun	3.44	4.03	3.44	3.43
IO, Stop	4.39	4.70	4.30	4.31
Tube, Spun	3.80	4.04	3.80	3.43
Tube, Stop	4.39	4.70	4.30	4.31

G.3 OPEN

The time taken to open a file was measured by this test. There are three variants on this test, "In" (open for input only), "Out" (open for output only), and "Update" (open for input and output).

OPEN	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
In, Spun	0.01	0.40	0.00	0.01
In, Stop	0.50	0.75	0.59	0.54
Out, Spun	0.39	0.79	0.39	0.19
Out, Stop	0.91	1.15	0.99	0.74
Update, Spun	0.01	0.39	0.01	0.00
Update, Stop	0.50	0.74	0.58	0.55

G.4 BPUT 32K

This test measured the time taken to write &8000 bytes into a file, one at a time. There is one variant on this test, where the second processor's fast BPUT call was used.

BPUT 32K	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
Out	54.32	54.36	54.33	55.02
Up	76.04	75.75	76.28	76.03
Fast, Out	28.80	29.23	28.42	29.41
Fast, Update	49.92	49.75	50.46	50.00

G.5 BGET 32K

The time taken to read &8000 bytes from a file one at a time was measured by this test.

BGET 32K	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
Get	52.70	52.50	52.71	52.86

G.6 MOVE

This test measured how fast the sequential file pointer can be moved.

MOVE	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
Out	29.15	29.87	29.81	29.60
Up	29.80	29.87	29.80	29.26

G.7 GBPB

There were two variants of this test, where &8000 bytes were read or written to a file using the OSGBPB call.

GBPB	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
Read, IO	29.11	29.17	29.12	29.24
Read, Tube	29.14	29.17	29.11	29.24
Write, IO	29.28	29.40	29.28	29.27
Write, Tube	29.27	29.39	29.27	29.28

G.8 SHIFT

This test measured how fast the sequential file pointer could be moved while performing file operations.

MOVE2	HDFS root	HDFS sub-dir	HDFS acorn	DNFS 1.20
Read	0.40	0.48	0.40	0.39
Write	0.62	0.68	0.62	0.62